10-30-2020

Spam Review Detection Using Self-Organizing Maps and Convolutional Neural Networks

Ashraf Neisari
University of Windsor

Spam Review Detection Using Self-Organizing Maps and Convolutional Neural Networks

By

Ashraf Neisari

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2020

Spam Review Detection Using Self-Organizing Maps and Convolutional Neural Networks

by

Ashraf Neisari

APPROVED BY:

M. Azzouz
Department of Electrical and Computer Engineering

A. Jaekel
School of Computer Science

L. Rueda, Co-Advisor
School of Computer Science

S. Saad Ahmed, Co-Advisor
School of Computer Science

September, 9, 2020

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

Chapter 2 of the thesis was co-authored with Luis Rueda and Sherif Saad. S. Saad contributed with initial research area and its fundamentals. L. Rueda elaborated on the new novel approach implemented in this work. Everyone contributed in finalizing the idea and follow-up discussions. A. Neisari implemented the method, data preprocessing, experimental design, and the data analysis. A. Neisari and L. Rueda wrote the contents of the chapter. L. Rueda and S. Saad assisted in reviewing the content of the chapter.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis. I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes an original paper that has been previously submitted for publication in the following IEEE journal:

Thesis chapter	**Publication title**	**Publication Status**
Chapter 2	Ashraf Neisari, Luis Rueda and Sherif Saad, Spam Review Detection Using Self-organizing Maps and Convolutional Neural Networks, IEEE Transactions on Information Forensics and Security, 2020	Submitted

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution. I understand that my thesis may be made electronically available to the public.

ABSTRACT

Online public reviews have significant influenced customers who purchase products or seek services. Fake reviews are posted online to promote or demote targeted products or reputation of the organizations and businesses. Spam review detection has been the focus of many researchers in recent years. As the online services have been growing rapidly, the importance of the issue is ever increasing and needs to be addressed properly. In this regard, there is a variety of approaches that have been introduced to distinguish truthful reviews from the fake ones. The main features engineered in the past studies typically involve two types of linguistic-based and behavioural-based characteristics of the reviews. Unsupervised, supervised and semi-supervised machine learning methods have been widely utilized to perform such a classification.

This work introduces a novel technique to detect fake reviews from the genuine ones using linguistic features. Unsupervised learning via self-organizing maps (SOM) in conjunction with a convolutional neural networks (CNN) are employed to perform classification of the reviews. We transform the reviews into images by arranging semantically-similar words around a pixel of the image or equivalently a SOM grid cell. The resulting review images are consequently fed to the CNN for supervised training and then classification. Comprehensive tests on two gold-standard datasets show the effectiveness of the proposed method on single and multi-domain contexts. Observing our results, we deducted that using GloVe 300-dimensional embedding and higher resolution SOM grid maps, our method achieves very good results.

DEDICATION

This thesis is dedicated to the loves of my life, my husband, Farshad, and my daughter, Nika, who are my inspirations in life.

AKNOWLEDGEMENTS

I would like to thank all the people who made this work possible, especially my supervisor Prof. Luis Rueda and co-supervisor Prof. Ahmed Sherif Saad.

I would like to thank professor Luis Rueda for his exceptional support and for providing great ideas that were used in this research. I learned a lot of new and practical topics from him, in his machine learning course, and also throughout my study. His enthusiasm, knowledge and close attention to detail have been an inspiration and kept my work on track. He always motivated me to pursue and take more advanced courses at the university and do not hold back if times seemed to be difficult. He was always patient, supportive, understanding and most noticeably extremely responsive when I was stuck or had questions.

I am also grateful for the generous support and valuable contribution of my co-supervisor, professor Sherif Saad Ahmed. He would always provide practical suggestions and insightful comments. He was a great help, especially in suggesting and narrowing down on selecting my thesis topic. I also had the chance to take his advanced security course and grasp very valuable knowledge and up to date topics in that course. He always keeps everyone motivated and challenges his students to keep them focused. He makes sure that everyone's needs and concerns are always heard.

I would also like to thank Prof. Arunita Jaekel and Prof. Maher Azzouz, my thesis internal and external examiner committee members. Their great comments during my thesis proposal challenged and inspired me to stay focused and examine different aspects in my research area for better outcomes.

Finally, It is been an honor to work with my dear colleagues in my lab, Alexandru Filip, Nazia Fatima, AbedlrahmanAl-Khateeb, Osama Hamzeh. Specially it is with pleasure that I acknowledge the guidance of my amazing lab mate Alexandru Filip. His great tips based on his deep knowledge of Tensorflow libraries helped me a lot when I was stuck with specific implementations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
GloVe	Global Vectors
TF-IDF	Term Frequency - Inverse Document Frequency
CBoW	Continuous Bag-of-Words
AI	Artificial Intelligence
ML	Machine Learning
NN	Neural Network
DL	Deep Learning
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
SOM	Self-organizing Maps
BMU	Best Matching Unit
NLP	Natural Language Processing
API	Application Programming Interface

CHAPTER 1

Introduction

1.1 Introduction to Spam Reviews and Detection

Online reviews are important motivators for shaping opinion and consequently making decisions. Increased sales revenues can, therefore, be achieved through positive reviews while negative ones can significantly damage the reputation and sales. Through monitoring online reviews and opinions, organizations can anticipate people demands or desires and their willingness on purchasing specific types of products. Companies and retailers conduct various research studies on customers purchasing behaviour and opinion to adjust and promote their production and marketing strategies. Unfortunately, not all of the posted reviews are genuine as some wrongdoers are hired to write undeserving positive or negative reviews about a product or service. Thus, in recent years, fake review detection has gained significant importance for different organizations and companies in such a way that they have been dealing with it in various ways [1].

Also, some companies try to buy the customers endorsement in different ways and unfairly manipulate their reputation. For instance, one clinic in the United States [6] sent postcards to its patients indicating they will give the patients a Starbuck gift in return if they give 3+ star reviews to the clinic on Yelp, Facebook or Google.

In general, there are three categories of fake reviews as described by Jindal et al. [7]. First, false opinion that mislead readers by undeserving positive reviews (hyper spam) or negative reviews (defaming spam). Second, reviews that target a brand only, but not specific products. Third category includes non-reviews, the ones

categorized as either advertisements or irrelevant texts. It is shown by various studies [7] that identifying fraudulent reviews in the second and third categories are easier than the first one. This is due to the fact that those categories can be recognized by experienced users manually. For the first category a variety of methods, including those using Artificial Intelligence (AI) algorithms, have been proposed but non-of them have thoroughly addressed the open questions in this field yet. Therefore, untruthful review detection is still a challenging open subject, requiring more robust solutions.

There are many obstacles for detecting the first type of opinion spam. For instance, there is no baseline by which the proposed methods can be effectively compared to in order to evaluate the performance of other methods. For instance, lack of a true representative real-world dataset separating fake reviews from the real ones makes it difficult to estimate the accuracy of the methods suggested. All the datasets used are somehow synthetic. Some researchers [10, 12, 11, 4], however, employed experts to manually label very small sections of their dataset based on predefined characteristics of spam reviews or spammers behavior. At the same time, even though most of the AI algorithms introduced are not sufficiently effective for automatic detection of review spams, they are still more reliable than manual detection.

New challenges have also risen as spammers use advanced AI algorithms to write fake reviews which sometimes are hardly distinguishable from the truthful ones [17]. These kinds of spammer's attacks on online review services reduce the challenge and costs of hiring human workers to write fake reviews. Producing cheep and scalable fake reviews can be very attractive for wrongdoers since they can easily control the sentiment of the reviews for products or services. Yao et al. [19] identified a new class of attacks to online review systems which uses deep learning algorithm. They employed Recurrent Neural Networks (RNN) to automate the generation of fake reviews that are difficult to be recognized as fake or truthful. Below are three reviews which were composed by AI bots about a restaurant in New York [17].

1) "I love this place. I have been going here for years and it is a great place to hang out with friends and family. I love the food and service. I have never had a bad

experience when I am there."

2) "I had the grilled veggie burger with fries!!!! Ohhhh and taste. Omgggg! Very flavorful! It was so delicious that I didn't spell it!!"

3) "My family and I are huge fans of this place. The staff is super nice and the food is great. The chicken is very good and the garlic sauce is perfect. Ice cream topped with fruit is delicious too. Highly recommended!"

This shows how challenging it can be to distinguish fake reviews from the real ones. The bots which composed those reviews were trained by neural networks (NN) using reviews from Yelp dataset. As researchers introduce new methods using advanced algorithms such as NN and deep learning (DL), spammers are also finding new ways to hide their malicious activities. As such, there will be arm-race in the future between bots constructing fake reviews using complex AI algorithms and bots detecting them using similarly advanced algorithms.

1.2 Problem Statement

Reviews have significant influence on customers for selecting or buying services or products. Keeping the reviews as a valuable and reliable source of information for both customers and organizations, untainted with the fake reviews, is then very important. As such, there is a need to detect and prevent fake reviews propagation to help customers and businesses being shielded from the fraudulent information.

The problem can be informally stated as follows. Given a set of reviews, fake or truthful, design a machine-learning-based mechanism that can identify spam reviews from the ham ones based on textual features.

1.3 Motivation

Due to significant rate of posted fake reviews there is a need for automated intelligent methods to filter them out to prevent misleading the online shoppers. This is so much needed that all the big companies such as Amazon, Facebook and Yelp have

been implementing methods to detect fake reviews. To date, a number of researchers have focused on detecting fake reviews and proposed a variety of methods. However, there is still room to come up with and evaluate different methods to detect fake reviews with potentially better performance.

1.4 Machine Learning

Machine Learning is a branch of AI aiming to recognize patterns in the raw data [15]. The goal of ML is to automatically learn without being explicitly programmed and to predict new inputs using previous examples and learned patterns. The performance of the ML algorithms adaptively improves as the number of samples for the training and learning phase increases. In recent years, ML has become more popular than before due to the ever-growing volume of data, computational power and storage capacity. ML can be a perfect tool to solve complex problems involving large volumes of data with many variables. For instance, handwriting, image recognition, predicting shopping trends, spam email detection, speech recognition, and credit scoring are good examples suitable to be approached using ML. ML methods can be categorized in three types: supervised, unsupervised and semi-supervised.

Supervised ML algorithms are widely used to detect future events by being trained sufficiently on the past labeled events. During the training phase the algorithm compares its current predictions with the true labels from the dataset and calculates the error. Then, the algorithms try to reduce the overall error by changing the hyperparameters. This process is iteratively applied until an optimized set of weights for a minimal error is achieved. Supervised learning is broadly used in two types of applications; classification, which predicts discrete outputs, and regression, which predicts continuous outputs.

Unsupervised ML algorithms, in contrast to the supervised ones, are used where there is no classified or pre-existing labeled samples. These types of algorithms detect hidden patterns of the data and use them to group and cluster them accordingly. Unsupervised learning is a good option to find internal representations and use them

to divide the data into multiple clusters.

Semi-supervised ML algorithms are used when there is only a small amount of labeled data and a large volume of unlabeled data. Since labeling the datasets manually is an expensive and daunting task, unsupervised learning can be used to label a massive amount of data leveraging a small amount of manually labelled data.

1.4.1 Neural Networks

Inspired by the structure of the animals and human brains, artificial neural networks (ANNs) or simply NNs are complex processing algorithms based on simple elements. Human brain as an extremely powerful data processor, similar to other organs, is made up of specialized cells [5]. These cells, called neurons, process data extremely simpler than the whole brain. The neurons are interconnected to other ones using signal receptors called dendrites, and signal emitters called axons as shown in Figure 1.4.1.

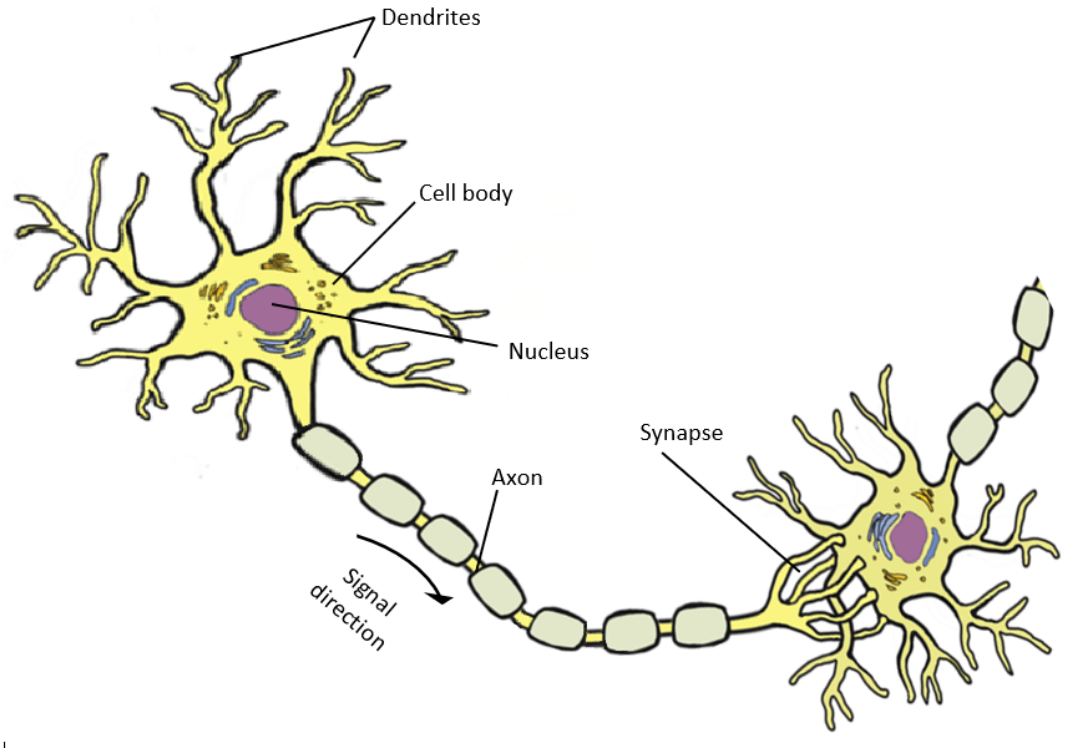

Fig. 1.4.1: Two connected biological neurons.

Each neuron has a cell body, many receptors and one axon which later is divided into two branches. Data is passed through and processed by the neurons in the form of physical electrical pulses. Fixed and relatively simple processing of input data is done inside the cell body while the interconnection and passing through the signals between neurons happens in regions called synapses. Synapses are where learning happens. Extreme processing power of the brain using simple processing power of neurons is known to come from using many slightly different types of neurons, in many layers, and training them.

Artificial neurons, thereafter referred to as neurons in this work, and their networks, NN, were then inspired by and envisioned to mimic the biologic ones. Each neuron, as depicted in Figure 1.4.2 has many inputs $x_1, x_2, ..., x_n$, which are multiplied by some weights $w_1, w_2, ..., w_n$, and then a non-linear activation function f is applied on the some to generate its only one output y.

$$Y = f(\sum_i W_i X_i) \tag{1}$$

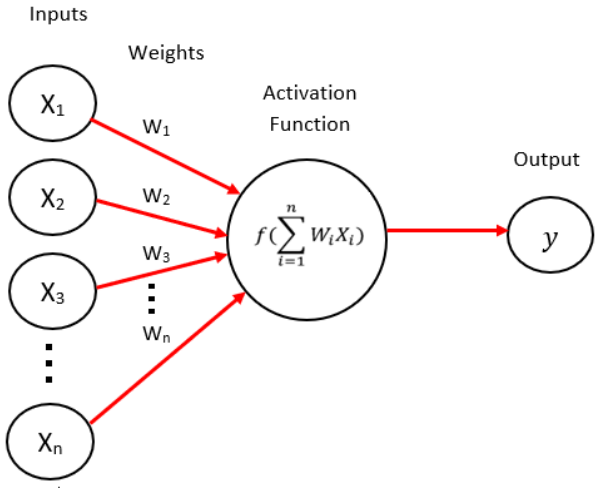

Fig. 1.4.2: An artificial neuron. The neuron sums up the weighted inputs which is then passed through an activation function.

The activation function can be of many types of Sigmoid, Tanh, ReLU, etc. [2] as noted below.

$$\textbf{Sigmoid} \qquad f(x) = \frac{1}{1+e^{-x}}\,, \tag{2}$$

$$\textbf{tanh} \qquad f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\,, \tag{3}$$

$$\textbf{ReLU} \qquad f(x) = max(0,x)\,, \tag{4}$$

$$\textbf{Leaky ReLU} \qquad f(x) = max(0.1x,x)\,. \tag{5}$$

Different activation functions are the main factor for distinguishing one neuron from a neuron of another type. Many similar neurons can be arranged in parallel to make up a layer of neurons. Layers of neurons can be arranged into a series to make up a network of neurons, a NN. Layers of multilayered networks can be distinguished as the input layer, the output layer, and the layers in between, referred to as *hidden layers*. The number of hidden layers is called *network depth*. The input layer feeds the neurons of the first hidden layer while the neuron in other layers are activated by other neurons in the previous layer through their interconnections.

The performance of any predictor is often represented by a loss function, which is a function of the predictor parameters. The loss function can be optimized for the predictor's best performance. Multilayer NNs contain many computational layers and so the loss function for the whole network is a complex function of the weights in all the layers. The loss function gradient is then used to minimize the loss through updating the network weights in a process called backpropagation. Network training is done by many iterations of first calculating the network outputs for different inputs, and then updating the weights through backpropagation. Thus first, in the forward phase the network output is computed based on the input values and the current weights. The actual output values are then compared to the computed ones. The gradient of the loss function is calculated for each weight in all the branches of the network by implementing the chain rule. The weights are then updated as follows:

$$W(t+1) \quad = \quad W(t) \quad - \quad \alpha \quad \frac{\partial L}{\partial W}\,, \tag{6}$$

where α is the learning rate, a hyperparameter to be set.

A well trained NN can then be used to predict or classify the outcome for a new set of inputs with good accuracy, a procedure called inference. Inference is a straight forward set of calculations which can be done using a modern cell phone processor. Unlike inference, training of a NN is a cumbersome repetition of lengthy calculations and thus much more time consuming. Therefore, training would need considerable hardware setup including arrays of powerful graphical processor units, GPUs, depending on the complexity of the problem and the size of the training data. On the other hand, once the training is done successfully, hopefully ,in no more than a couple of iterations, the inference can be preformed relatively easy as many times as needed.

1.4.2 Deep Learning

Designing a NN that has more layers with increased units of neurons at each layer leads to a new framework of deep neural networks or simply the DL paradigm [3]. DL requires a sufficiently large model and very large datasets for the training phase.

Deep learning, as a sub-field of ML, attempts to automatically learn useful features or representations from the dataset. This is in contrast to conventional ML algorithms which work well due to human-designed representations and input features. Deep learning algorithms are designed to learn multiple levels of representations which sometimes is even hard for the human brain to detect. In other words, in ML, the difficult job of defining input features is done by humans, whereas in deep learning methods these features are learned automatically.

1.4.3 Convolutional Neural Networks (CNN)

As a relatively new twist to the traditional NN and inspired by the cognitive neuroscience, CNN were developed as a type of the deep neural networks. One of the early

studies performed in this area by Hubel and Wiesel's was on the visual cortex of cat's brain [18]. They discovered that the visual cortex is largely made up of simple neurons, which respond to small motifs and complex ones that respond to bigger motifs. Later works in this area led to the design of the CNN framework which works with grid-structured inputs.

Two-dimensional images as the most common grid-structured inputs with intense spatial dependencies can therefore be successfully analyzed using CNNs. Grayscale images are two-dimensional grids of pixels intensities, while color images can be represented by three grids of intensities for the three principal colors of red, green and blue, or RGB. A typical process for the classification of images is depicted in Figure 1.4.3.

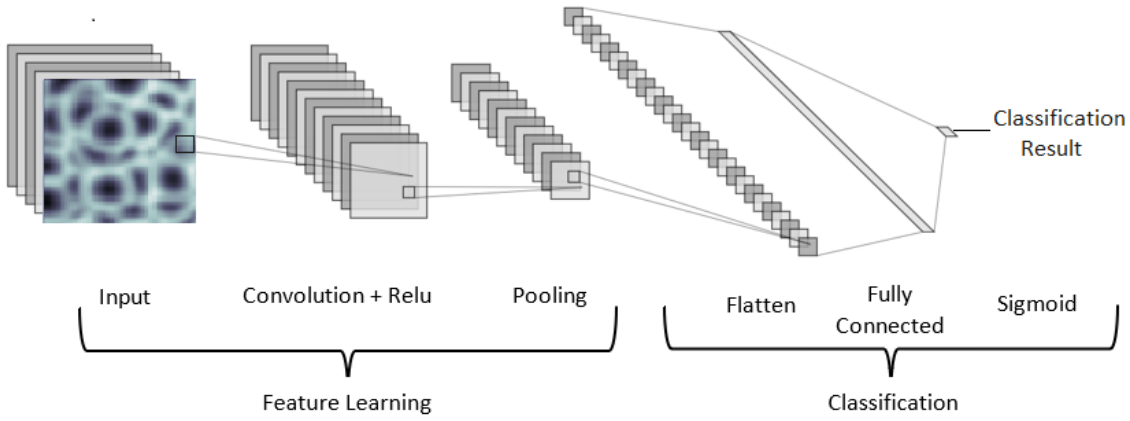

Fig. 1.4.3: A typical CNN used for image classification.

Convolution is first performed on all regions of the image by the application of filters of usually small sizes, 3×3 for instance, to detect small features in the image. Activation of often ReLU type operates on the convolution output and the result of this process is then called activation map. It is basically revealed in the activation map if the pattern of that filter exists in any region of the image. **Pooling** is then performed to reduce the size of the activation maps resulted from a convolutional layer. This is very important as the size of large images can lead to shear amount of resources needed for processing. This basically determines if simple features exist in larger regions of the picture or not.

Multiple consecutive layers of convolution and pooling determine the existence of more complex features in the image as it goes through further processing. The size and number of these layers, the hyper-parameters, are selected iteratively based on the performance of the network. **Fully connected** NN layers can then be used to train and learn the relation of those complex features in the image and output the probability for the class of object the image represents. These layers are described in more detail below.

1.4.3.1 Convolutional Layer

As mentioned earlier, the role of a convolutional layer is to detect a feature in the grid data it is operating on, as the network input image, or the activation map output of the previous convolutional layer. The feature that a convolution is trying to detect in an image is called filter or kernel. A filter is often a small square matrix of 3×3 or 5×5, for instance. In addition, there is also the depth of the filter, equivalent to the number of layers of the input. For instance, a $3 \times 3 \times 3$ filter can be applied to an RGB color image.

These filters are initially set randomly, and then updated through iterations to best represent the detected features of the inputs. This is where the magic of the CNN happens and the hidden input representations are discovered automatically through the CNN process. This is also why in contrast to the traditional NN, there is no need for human-provided features.

The convolution of a 4×4 black and white binary image using a 3×3 filter is depicted in Figure 1.4.4 for illustration. The operation puts the filter over every possible position of the image for the sliding of the filter one pixel at a time, or the stride of one. For the selected window of the image at a time, the result of dot product between the values of all filter cells and the corresponding image window cells are calculated. This is similar to mathematical convolution operation and that is the reason this type of layers is called convolutional. Consequently, an activation of often ReLU type is applied to the result of the convolution and the output is stored in a grid cell.

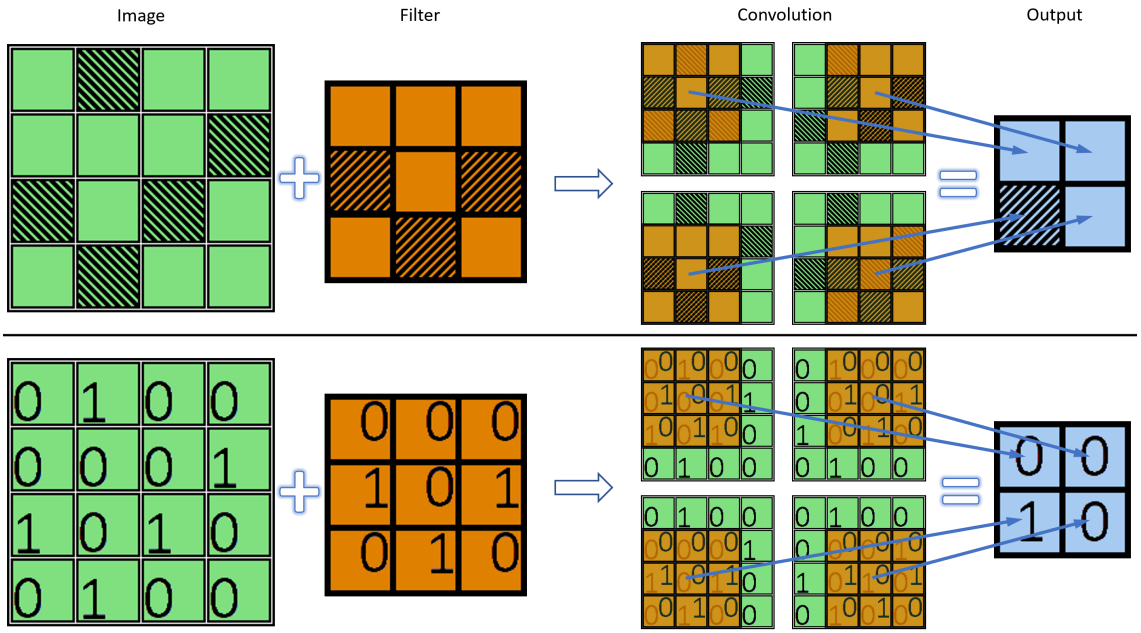

Fig. 1.4.4: Convolution of a binary image with a binary filter. Top schematic, bottom numerical representations.

From a different perspective, each cell of a filter can be considered as a weight. The convolution operation involves the summation of weighted inputs and the application of an activation function. This is the same operation performed in artificial neurons as explained earlier. In other words, a filter can be viewed as a neuron operating on the input values coming from a small window of the image. As the filter slides over all possible regions of the image, the effect is equivalent to having a vast grid of parallel neurons for the filter operating on all small windows of the input image. Therefore, the output of all these neurons are stored in the cells of an output grid, the activation map, with the exact size of all the neurons used for that filter. All these neurons share the same weights, a concept called parameter sharing. It should be noted here that all these neurons of the filter are only connected to a small window of the input image. Thus, these neurons are locally connected, and the whole layer is not considered as a fully connected one.

A convolutional layer can have not just one, but many filters, which results in many layers of activation maps for the same convolution operation. Also, for all channels of an input, such as an image with three layers of RGB colors, a filter operates on all

the layers at the same time. This adds another dimension to a filter as stated earlier, three for an RGB image, compared to two for a gray-scale one.

The convolution operation of a filter on an image, or an activation map, yields another activation map of slightly smaller size, depending on the filter size and the stride. Many different filters of the same size are usually used in a single convolutional layer, and so as many output activation maps will be the output of just one convolutional layer.

To put this into the right context, consider an RGB color image of just $64 \times 64 \times 3$ pixels with 10 filters of $3 \times 3 \times 3$. Such a convolutional layer results in a 10-layer activation map of 62×62 pixels as the output. This is more than three-fold increase in data to be processed in the next layers. The numbers for this example was kept very low intentionally to show the magnification. Imagine using hundreds of filters from the millions of possible ones, combined with the images of FHD (1920×1080) quality coming from sensors available on average cell phones these days to grasp the shear amount of data needed to be processed, and that is only for the still images. In contrast, 60 frame per second video streams are the norm and minimal for today's applications. This shows a mechanism to reduce the amount of data to be processed in the next layers is not only very useful, but also absolutely crucial. That is the role of a pooling layer used immediately after a convolutional one, otherwise, the output activation maps from one convolutional layer could be fed directly to another convolutional one to extract more complex features.

1.4.3.2 Pooling Layer

As stated before, pooling is a mechanism that reduces the amount of output data coming from a convolutional layer. A pooling layer hence makes the representations smaller and more manageable. Similar to the convolutional layer, there is a window in a pooling layer, sliding over the activation maps resulted from the convolutional layer. For the pooling, this window operates on each layer of the activation maps independently. Therefore, unlike in the convolution operation, the number of layers, activation maps, or the depth does not change after a pooling operation. The pooling

operation on a layer of activation maps is shown in Figure 1.4.5.

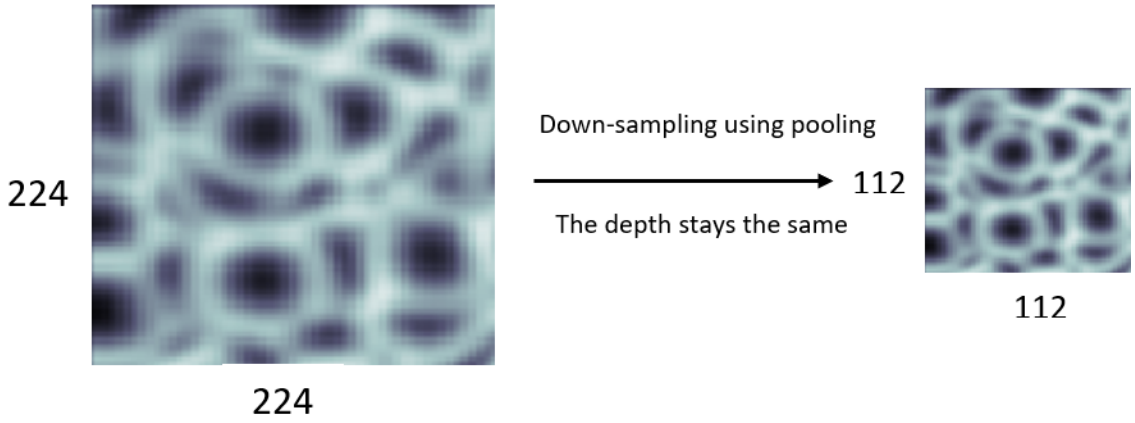

Fig. 1.4.5: A 2×2 pooling operation with a stride of 2 and its effect on the size of a feature map.

There are a couple of pooling methods available including max pooling and average pooling. Although average pooling performs better in some applications, max pooling was found to outperform all the other methods in most applications. Max pooling as shown in Figure 1.4.6 involves picking the maximum of all the cell values in the sliding window as the output of the operation.

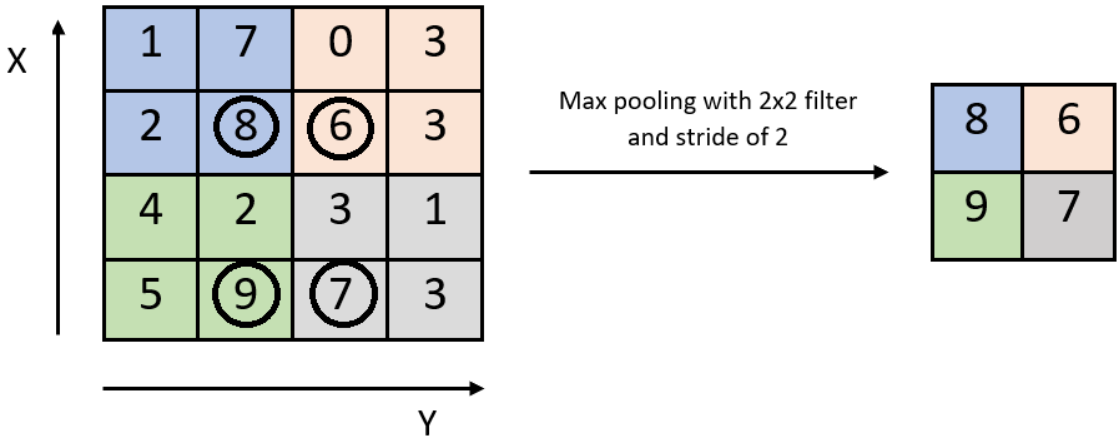

Fig. 1.4.6: A 2×2 max pooling operation with a stride of 2.

1.4.3.3 Fully Connected Layer

We have a mechanism so far to detect simple to complex features of an image, or a data grid, through a number of convolutional layers. We also came up with the

pooling technique that makes the size of activation maps resulting from each convolutional layer smaller and more manageable. The last step is to design a mechanism for interpretation of an image and its classification, based on the features already detected. This can be done using a fully connected multi-layer NN. A multi-layer NN is usually used with the sigmoid or softmax activation function in the last layer for the final classification. A variety of activation functions such as ReLu, tanh, Maxout, Sigmoid, are also used on the first layers depending on the problem type. Unlike the convolutional layers, neurons of these layers are connected to all outputs of the previous layers. These layers are called fully connected layers, therefore. Also, since the output of these layers are used to predict known objects, the fully connected layers input are designed to be one dimensional. As such, in practice, there is a flattening operation at the end of the final pooling layer to convert multi-dimensional activation maps to one dimension suitable for the fully connected layers. Figure 1.4.7 illustrates a sample of fully connected NNs.

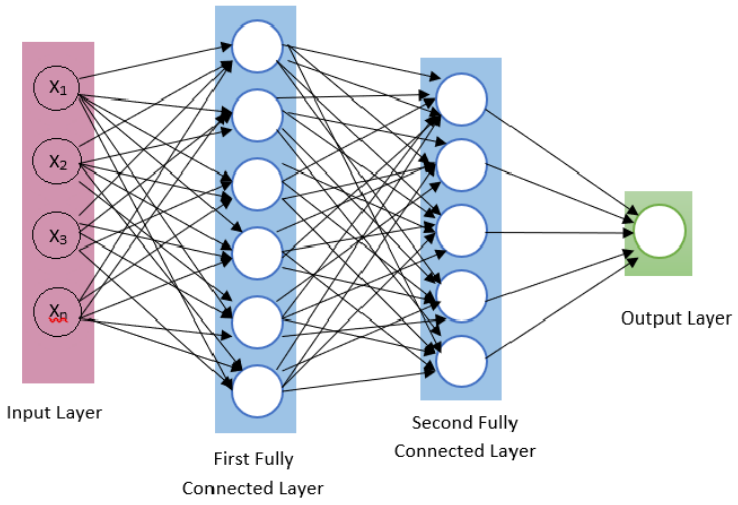

Fig. 1.4.7: A sample of fully connected NNs with two hidden layers.

1.4.4 Self-organizing Maps

A self-organizing Map (SOM) is an unsupervised NN that is mostly used for projecting high-dimensional data points onto a lower-dimensional space, typically, two-dimensional [16]. Unlike other ANNs, which employ backpropagation, through gra-

dient descent and iterative error reduction, the SOM is a competitive learning mechanism. During the learning phase each node or neuron competes with other ones to become closer to the input data points. At the end, a map is constructed in such a way that similar input data points are clustered and grouped together.

A SOM creates a low-dimensional matrix, a grid map, in which each cell is considered as a neuron. Each neuron has a weight vector of the same size as any of the input data points. This matrix is used to evaluate the distance of any input vector in the dataset from the weights of each cell.

The weights assigned to each cell is randomly initialized at the start of the training phase. Then, iteratively and for each data point, the closest neuron, the one with the smallest distance to the data point, is found and the data point is assigned to. This neuron is referred to as the Best Matching Unit (BMU). Next, the BMU weights are updated in a way that the neuron is shifted towards the data point. The weights of BMU neighbouring neurons are also updated during the same iteration and thus they are shifted towards the same direction as the shifted BMU, but with a smaller rate. The amount of changes in the neuron weights decreases as more iterations are performed during the training, as the grid map of neurons becomes closer and closer to the input data points during every iteration. The process proceeds with all the data points during the training. At the end of the training phase, each data point is assigned to a cell of the SOM grid map. Similar inputs are grouped together around neighboring cells. For instance, Figure 1.4.8 depicts the world poverty grid map constructed by SOM based on 39 quality of life factors such as education, nutrition, and health, among others, of different countries [13].

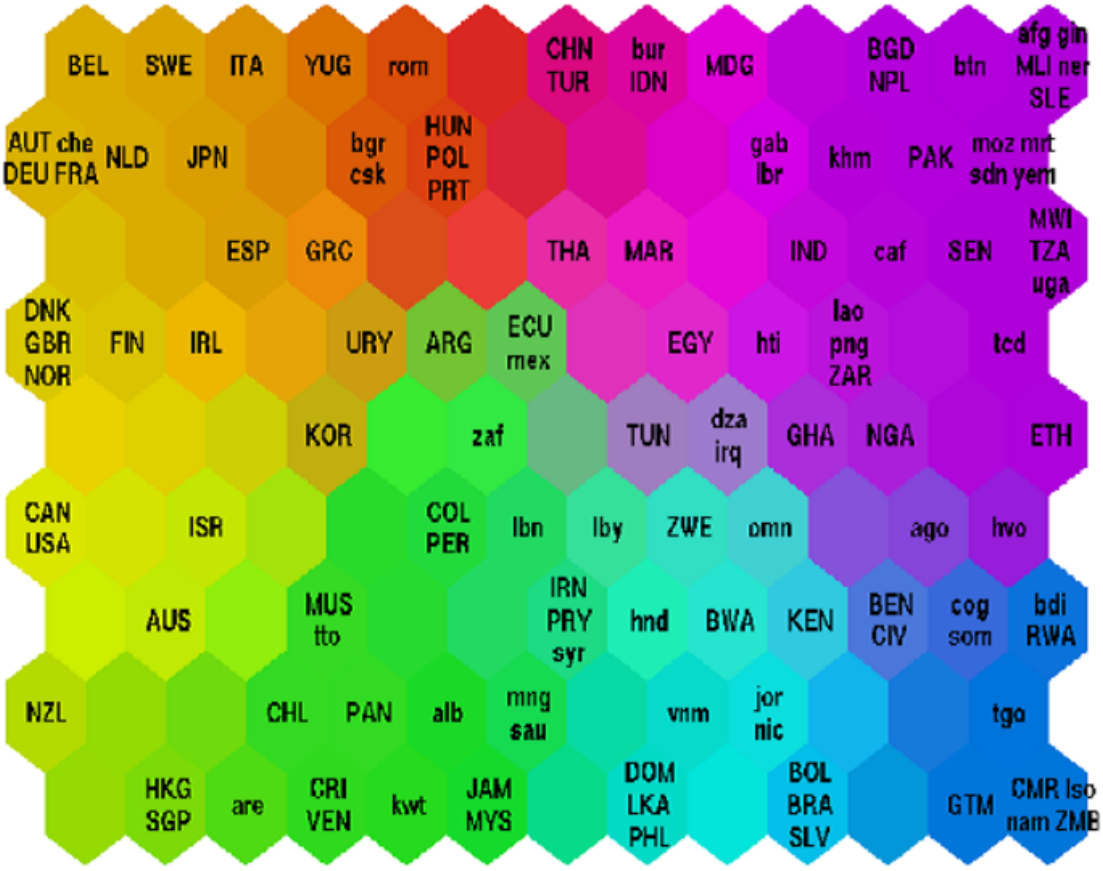

Fig. 1.4.8: The SOM map clustering of the countries based on their poverty levels [13]. Countries are clustered using different colors where those colors illustrate the poverty type.

Countries that have similar values for various life factors found a place near each other on the map. The different clusters on the map were encoded with different colors, nevertheless those colors change smoothly over the map. As a result of the process, each country was assigned a color describing its poverty type in relation to other countries. For instance Japan was clustered close to European countries such as Netherlands, Sweden and Italy for having similar quality of life. Or for the quality factors on the other side of the spectrum countries such as Pakistan and Yemen in south Asia and Middle East were put close to African countries such as Sudan and Mozambique.

1.5 Word Embedding

Word embedding consist of techniques that are used to represent unstructured data about words as fixed dimension vectors of real numbers. The goal is to capture the essence and relationship among the words automatically. In Natural Language Processing (NLP), capturing words relationships and similarities can improve the performance of some applications such as question answering systems, information retrieval, and machine translation, among others. While there are many methods to perform word embedding, the most popular ones are Word2Vec [8] and GloVe [14]. It should be mentioned that word embedding as vectors can undergo all vector operations including vector summation, inner product. Also, similar to other vectors, closeness or similarity of two vectors, x and y can be represented by the cosine of the angle between the two vectors as follows:

$$Similarity(x, y) = \cos(\theta) = \frac{x \cdot y}{\| x\| \cdot \| y\|} . \tag{7}$$

As such, the objective is to group related words into close spatial positions. Mathematically speaking, for a good embedding technique, the cosine value of the angle between the vectors for similar words, as described in the formula, should be close to one.

1.5.1 Word2Vec

Word2Vec is a technique used to learn high quality word representations from large data sets containing billions of words. The method was first proposed by Mikolove et al. [8]. The resulting word vectors capture multiple types of semantic and syntactic word relationships. For instance, it is shown that vector("King") - vector("Man") + vector("Woman") results in a vector that is (cosine) closest to the vector representation of the word "Queen" [9]. Also, the word "France" is similar, or close, to "Italy" and other countries. "France" also has semantic relationship to "Paris" in the same sense "Germany" has to "Berlin". Furthermore, the word "big" is similar to "bigger", and "small" is similar to "smaller", in the same sense.

There are two Word2Vec architecture types for learning distributed representations of words as shown in the Figure 1.5.1, CBoW and Skip-gram [9]. Continuous Bag-of-Words model (CBoW) predicts the probability of a center word given the surrounding context words in a predefined window. Skip-gram, on the other hand, tries to predict the context words based on the center word given a defined window. Comparing the two, CBoW is faster and learns better representations for more frequent words, whereas Skip-gram performs better with small amount of data and rare words.

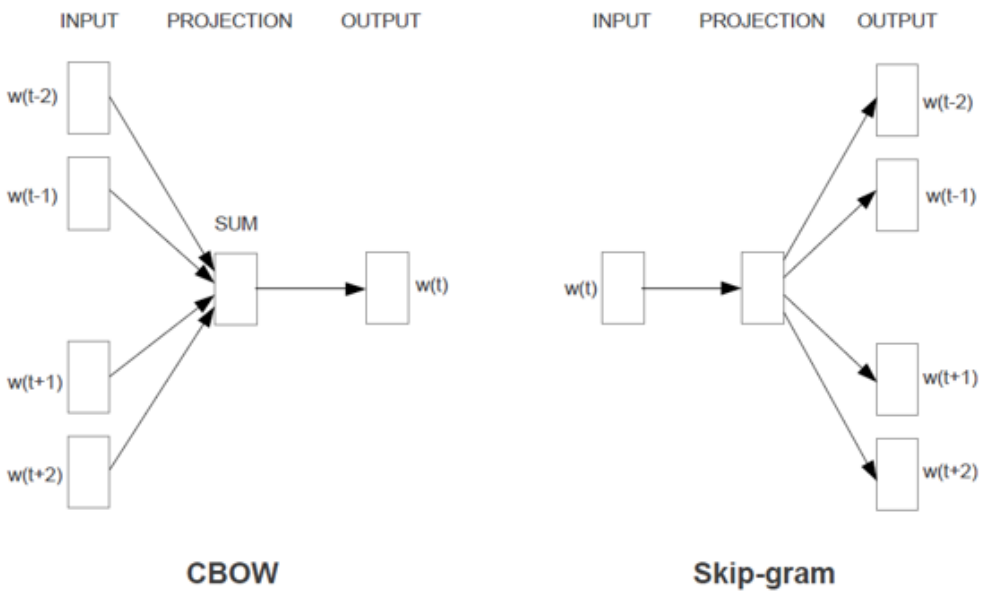

Fig. 1.5.1: Schematic representation of the two model architectures of Word2Vec [9].

The process of Skip-gram is shown below for illustration. The core idea is that the meaning of a word is given by the words that frequently appear nearby. While training, Skip-gram uses many given context words appeared close to the center word, to build up a representation for. This is depicted in Figure 1.5.2 for the center word "algorithms".

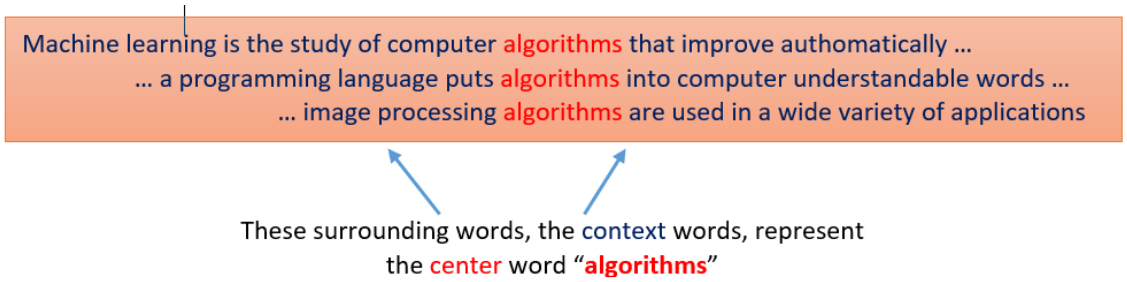

Fig. 1.5.2: An example of a center word and its surrounding context words used for skip-gram training.

The process for the learning phase starts by collecting a large amount of text for the corpus. Each unique word in the corpus is assigned a vector of fixed size and randomly initialized. Then, it goes through the corpus and zooms on a window of size m at a time. For each position t in the corpus, $t = 1, ..., T$, there is a center word and $m - 1$ context words as shown in Figure 1.5.3, where T is the number of words in the corpus. Using the similarity of the word vectors for the center and context words, their co-occurrence probability is calculated along the process. The vector representations of the words are then determined through optimizing a defined objective function for the prediction accuracy.

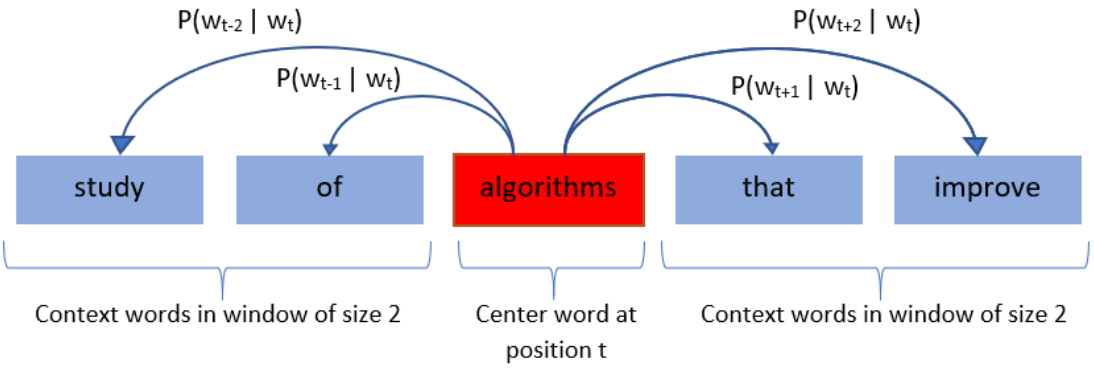

Fig. 1.5.3: The center word "algorithms" and its four surrounding "context" words within the window size equal to 5.

1.5.2 GloVe

GloVe stands for "Global Vectors" and is another powerful word embedding method that represents words as numeric vectors [14]. Similar to Word2Vec, GloVe assigns

semantically related words to close vectors, and unrelated ones to the vectors that are far from each other. Unlike Word2Vec, though, GloVe does not rely only on the local information, the ones surrounding the words. Instead, GloVe incorporates both local and global word relationships in a given corpus to capture their semantic similarities and assign vectors. GloVe is built around an important idea that co-occurrence matrix can be used to capture these relationships. The co-occurrence matrix of a corpus that contains V words is a $V \times V$ matrix. For instance, the co-occurrence matrix for the following corpus is shown in the Table 1.5.1.

"I like computer games."

"I like coding."

"I play basketball."

Count	**I**	**like**	**play**	**computer**	**games**	**coding**	**basketball**	**.**
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
play	1	0	0	0	0	0	1	0
computer	0	1	0	0	1	0	0	0
games	0	0	0	1	0	0	0	1
coding	0	1	0	0	0	0	0	1
basketball	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Table 1.5.1: The co-occurrence matrix for the sample corpus.

The element at the i^{th} row and j^{th} column, m_{ij}, denotes how many times the word i has co-occurred with the word j. Here, the stages for determination of the word vector representation are the same as those for Word2Vec. This time though, the determination is done with different probabilities and a different objective function.

1.6 Performance Metrics

The performance of any prediction method, ML based or otherwise, needs to be quantified in order to be universally comprehended and compared to other methods. In order to be able to measure the performance, first, there should be a set of data with the actual outcomes to be compared with the prediction method outcomes. This set reserved for the comparison is called test or validation data.

For a binary classification problem, depending on where a performance measure is originated from, medical diagnosis, for instance, different performance metrics have been introduced and preferred. Despite that, all metrics use the following assumptions to define a performance metric. A positive outcome is considered to be the case when a problem, condition or disease exists, or predicted to exist. A negative outcome, on the other hand, is when a problem, condition or disease does not exist, or predicted not to exist. Detecting a tumor in a medical image of an organ is considered a positive outcome, for instance, similar to detecting an email as a spam.

True positive (TP) prediction is defined as the situation where the true outcome was predicted and was confirmed by the actual true outcome. The same analogy can be used for defining **false positive** (FP), **true negative** (TN), and **false negative** (FN) situations. When prediction was the positive outcome but the outcome was actually negative, it is a FP situation. In a TN situation, a prediction of negative outcome was confirmed by the actual negative outcome. Finally, FN is defined as a prediction of negative outcome while the outcome was actually positive. These definitions are summarized in Figure 1.6.1.

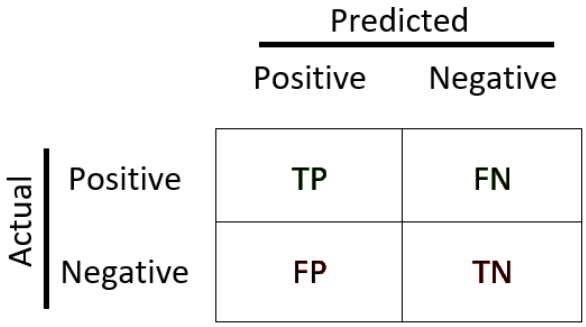

Fig. 1.6.1: The confusion matrix for the binary classification.

As the number of actual positive and negative cases are denoted by P and N, respectively, and the total number of cases as n, the following are intuitive:

$$P = TP + FN\,, \tag{8}$$

$$N = TN + FP\,, \tag{9}$$

$$n = P + N = TP + TN + FP + FN\,. \tag{10}$$

Based on these definitions a number of performance metrics can be defined as follows [15].

1.6.1 Accuracy

The ratio of the total number of correctly classified samples over the total number of samples is called accuracy. It can be mathematically defined for the binary classification problems as follows:

$$Accuracy = \frac{TP + TN}{n}\,. \tag{11}$$

1.6.2 Precision

Precision, Positive predicted value (PPV), or hit rate is the rate of correctly classified positives samples over all samples classified as positive. Therefore, for binary classification, it can be denoted as follows:

$$Precision = \frac{TP}{TP + FP}\,. \tag{12}$$

1.6.3 Recall

Unlike precision, recall is the rate of correctly classified positives samples over all true (actual) positive ones, whether they were classified correctly as positive, TP, or falsely classified as negative, FN. As such, recall can be denoted slightly differently

for the binary classification as follows:

$$Recall = \frac{TP}{TP + FN}\,. \tag{13}$$

1.6.4 F1-Score

Consider the case for a classifier with high FP and low FN numbers, or vice versa. For such a case, the precision and recall give considerably different answers for the performance of the classifier. Even accuracy would not provide the number it was designed for, since TP and TN would be imbalanced as a result of the total number of positive and negative samples, P and N, being imbalanced as well. F1-Score tries to assess a balanced performance metric by somehow finding a type of average between the precision and recall numbers, something resembling the geometric mean of two numbers. This is done by dividing the product of precision and recall by their average as follows:

$$F1\_Score = 2 \quad \frac{Precision \cdot Recall}{Precision + Recall}\,. \tag{14}$$

1.6.5 Cross validation

The weights of a trainable classifier and hence its performance measure can vary based on the, often, random initialization. In other words, a single experiment at times may result in a completely different good or bad performance for the same system. Therefore, training a classifier, or providing its performance measure for just one set of data can lead to a misleading performance metric.

m-fold cross validation divides the entire dataset into m disjoint sets of (almost) equal sizes. In this method, the classifier is trained m times. Each time one of the k sets is removed and kept as a test set, and the classifier is trained on the remaining $m-1$ sets. The model is then evaluated on the separated test set. For each iteration, TP, FP, TN and FN numbers are recorded and consequently averaged at the end of all iterations. The averaged numbers are used to calculate the performance of the model instead.

1.7 Proposed Method

This thesis proposes a new approach for the spam review detection problem. The approach starts with clustering the semantically related words presented in the review corpus in a grid map via constructing and training of a SOM. The grid map is then used to represent each review as a unique fixed-size image of multiple layers. Afterwards, the constructed review images are fed to a CNN for training and then classifying reviews as ham or spam. This approach utilizes linguistic features to extract lexical diversities spread in ham or spam reviews. The performance of this approach is also examined using well-known datasets for spam review detection. Details of this approach is described in Chapter 2.

1.7.1 Contributions

The contributions of this thesis can be summarized as follows:

- Proposed a new method for the spam review detection using SOM and CNN.
- Provided a mechanism to convert any review of variable length to a fixed length numerical array using a SOM.
- Envisioned and implemented the application of CNN to find the relations of the words in a lower-dimensional space.
- Developed a Python package for the proposed method, which is available at `https://github.com/Ashsari/spam_review_detection`.

References

[1] Jason Cohen. *How to Spot a Fake Review on Amazon*. `https://www.pcmag.com/how-to/how-to-spot-a-fake-review-on-amazon`. 2019.

[2] Yoav Goldberg. "Neural network methods for natural language processing". In: *Synthesis Lectures on Human Language Technologies* 10.1 (2017), pp. 1–309.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[4] Christopher Glenn Harris. "Detecting deceptive opinion spam using human computation". In: *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[5] Dominique A Heger. "An Introduction to Artificial Neural Networks (ANN)-Methods , Abstraction , and Usage". In: 2015.

[6] *How Yelp is weeding out fake reviews*. `https://www.cbsnews.com/news/yelp-behind-the-scenes-consumer-protections-eliminate-fake-reviews/`.

[7] Nitin Jindal and Bing Liu. "Analyzing and detecting review spam". In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE. 2007, pp. 547–552.

[8] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[9] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[10] Arjun Mukherjee, Bing Liu, and Natalie Glance. "Spotting fake reviewer groups in consumer reviews". In: *Proceedings of the 21st international conference on World Wide Web*. 2012, pp. 191–200.

[11] Myle Ott, Claire Cardie, and Jeffrey T Hancock. "Negative deceptive opinion spam". In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: human language technologies*. 2013, pp. 497–501.

[12] Myle Ott et al. "Finding deceptive opinion spam by any stretch of the imagination". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics. 2011, pp. 309–319.

[13] Kevin Pang. *Self-organizing Maps*. https://www.cs.hmc.edu/~kpang/nn/som.html#refer. 2013.

[14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[15] Duda Richard O, Hart Peter E, and Stork David G. *Pattern classification*. John Wiley & Sons, 2012.

[16] Giuseppe Vettigli. *MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map*. https://github.com/JustGlowing/minisom/.

[17] James Vincent. *AI trained on Yelp data writes fake restaurant reviews 'indistinguishable' from real deal*. URL: https://www.theverge.com/2017/8/31/16232180/ai-fake-reviews-yelp-amazon.

[18] Robert H Wurtz. "Recounting the impact of Hubel and Wiesel". In: *The Journal of physiology* 587.12 (2009), pp. 2817–2823.

[19] Yuanshun Yao et al. "Automated crowdturfing attacks and defenses in online review systems". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1143–1158.

CHAPTER 2

Spam Review Detection Using SOM and CNN

2.1 Introduction

The Internet has enormously grown in size and importance in the past few years, yet it has showed a huge and ever-growing impact on people's daily lives. People spend a significant part of their time surfing the internet to gain information on various topics, communicate with others, and read reviews, articles and news. The Internet also allows people to post reviews about different subjects based on their own knowledge and experience along with others' opinions viewed online. As such, they support or oppose different posts regarding products or services as well. Thus, online reviews play a significant role for both users and providers [22, 18].

Since anyone can freely post reviews without any limitations, wrongdoers can give undeserving positive or negative opinions to some targeted products, services and businesses. This is done, primarily, to promote or damage the reputation of the target. A person who posts fake reviews is called spammer and his or her posted reviews are called "spams reviews". We also call the reviews posted by a genuine customer, "ham reviews". Spammers are employed, some of them occasionally even by a competitor, to influence reputation, increase or decrease sales of a certain product. Products that show a higher rate of positive reviews are more appealing for the customers and may lead to increased financial gain for the producer [20]. On the other hand, products with dominant negative reviews may cause financial loss for the involved companies

[1, 27]. Therefore, truthfulness of posted opinions and reviews need to be examined to avoid misleading the public through deceptive information. In this regard, Ott et al. found in their tests that the average accuracy of three judges for detecting spam from ham reviews can be estimated to just 57.33% [16]. In addition, performing such a task manually is a daunting task. Therefore, using automatic detection of opinion spams utilizing state-of-the-art intelligent methods not only does the classification significantly faster, but it also allows to perform it much more accurately than a human expert.

In this work, we introduce a novel approach to distinguish fake reviews from the truthful ones. This approach uses self-organizing maps (SOM) and convolutional neural networks (CNN) in a special combination. We utilize both supervised and unsupervised learning to extract the hidden linguistic attributes from the reviews. After evaluating the performance of the method on a well-known hotel review dataset [16, 15] and a gold standard Multi-domain dataset [9], we compare and summarize its performance with other relevant methods. We also examine the effect of using a different prevalent embedding method of, Word2Vec [11] or GloVe [17] of 50, 100, 200 and 300 sizes, on the performance of the developed approach. We performed this by examining these kinds of effects with changes in the sub-components; The effect of choosing the SOM grid size and the neighborhood radius on the SOM unsupervised training. We then used the output of the SOM and feed them to the CNN classifier to distinguish spam reviews from truthful ones.

The rest of this chapter is organized as follows. Section 2.2 discusses the main approaches that have been utilized for spam review detection. In Section 2.3, we discuss the materials and methods used in this work, including the relevant datasets, data pre-processing and classification methods. Finally, We present the results in Section 2.4, along with a comparative analysis.

2.2 Literature Review

Spam review detection was first studied by Jindal et al. [7], who used the concept of duplicate and near-duplicate characteristics of the reviews. Since then, there has been a growing interest in this field, as fake reviews have become widespread and impacted various businesses. Due to the challenging nature of the fake review detection task, none of the research done so far has provided a robust solution to the problem, and there is still ongoing research to bridge the gap. Previous works can be categorized based on how different features are engineered and the classification methods utilized.

2.2.1 Feature Engineering

Different sources of datasets offer a variety of features available for classification of reviews as spam (fake) or ham (truthful). For instance, the TripAdvisor dataset does not contain characteristics of reviewers, timestamp, rating, and product ID, among others. This shortcoming makes it difficult for the researchers to apply a method devised for a specific dataset to other problems or datasets. Some compiled common features used by previous approaches can be categorized as being related to the review themselves or the characteristics of the reviewers.

2.2.1.1 Review Content

This category includes the body of the review related features, reviews content or contextual features. Some studies have used textual content and linguistics features, alone or in conjunction with other features, to assess the veracity of the review. Textual content features include language patterns, used terms, words meanings, and term frequencies, among others. Using textual content alone usually verifies fake reviews with moderate accuracy, e.g., 75% [10]. Spam identifiers utilizing linguistic features can often be fooled by wise spammers trying to write opinions similar to the genuine ones. In addition, textual features are domain-specific, which makes it difficult to create a unified method for cross-domain verification. For instance, the terminology used for describing a restaurant, such as “delicious” or “tasty”, can not

be used for car repair shops. Therefore, other features are often employed along with contextual analysis to boost the detection power and, subsequently, prediction accuracy.

Commonly used methods for feature extraction from relevant corpora include bag-of-words, n-gram, term frequency, semantic, sentiment, Linguistic Inquiry and Word Count (LIWC), Part of Speech (POS) Tagging, and Stylometric, among others. Asghar [13] extracted four types of contextual features, including unigrams, bigrams, trigrams and latent semantic indexing from the body of the reviews and fed them to different machine learning algorithms. They built sixteen models and among all those, logistic regression achieved the highest accuracy of 64%. Shahariar et al. [23] also selected features from the body of the reviews including n-gram, term frequency and word embedding. They reported very good results achieved using deep learning models. Their best reported prediction accuraccy is 94.565% using Long Short-Term Memory (LSTM) after splitting the data with a ratio of 70:30.

Saumya et al. [21] proposed a method using sentiment of the review in conjunction with other features for the classification. Their proposed method implementing Random Forest (RF) for classification resulted on 91% of F1-score. In a case study, Yilmaz et al. [26] also used the Doc2vec algorithm, which generates document embedding from the textual content of the reviews. This method addresses the issue of the reviews being in a variable length by generating a fixed-length embedding vector for each review to be fed to the classifiers.

2.2.1.2 Characteristics of the Reviewer

This subcategory includes information about the reviewer and group-related features. Users' behavior and footprint provide effective features for identifying fake reviews, spammers or spammer groups. These features, when used in combination with other features, have shown better results compared to employing linguistic or behavioral features alone. Compared to the methods that focus on contextual analysis, using the behavioral features reduces computational costs and saves time. The behavioral features used in the past studies include reviewers target products, rating, early posts

and ratings, ID, number of posted reviews and rate, review length, polarity mean and distribution, and content similarity, among others.

Mukherjee et al. [12] introduced a clustering approach by observing the behavioral footprint of the spammers and non-spammers. They hypothesized that these two clusters have different behavioral distributions including content similarity, reviewer maximum number of posted reviews, burstiness, ratio of first reviews, and early time frame, among others. Hussain et al. [6] used both linguistic and reviewer behavioral-based features in their proposed methods. They extracted thirteen different features based on reviewer behavior including review length, the ratio of first reviews, negative and positive reviews, activity window, review count, and others. They also used these behavioral based features to output a labeled dataset that groups reviews as ham or spam. They used this generated labeled dataset as the input to their second method for identifying spam reviews. In their second proposed method they extracted some linguistic based features and feed them to the same classification models used in the first method. The study showed behavioral based features achieve better results compared to using linguistic based features. Their proposed method using behavioral features yielded 93% accuracy and using linguistic features achieved 88.5% accuracy.

2.2.1.3 Metadata Features

This subcategory includes the review itself and product metadata related features. Some studies have utilized clues from metadata and considered reviews characteristics in general rather than just focusing on the content of each reviews. There are datasets that provide additional information needed for this approach. The metadata include price, sales rank, ratio of the positive to negative reviews, and target product ID. These features can help identify anomalous activities. The features of the metadata have been very beneficial, since they help recognize spammers and link them to various domains easily.

Names or IDs in metadata can be used to detect whether a specific product was the spammers' target. Spammers' groups can be spotted by examining the group of products being reviewed by a verified group of users in a short period of time [19].

Additionally, cross-domain spam detection can be achieved if spammers use the same IDs, under the same or different names, to review different targets. The number of feedbacks for the helpfulness of the products and percentage of feedback a review receives are useful in estimating the quality of the reviews [7, 4]. Having estimated the ratio of all good quality reviews to bad quality ones, and abnormal ratings of spammers reviews can be detected.

Some studies [2] have also used the location of the reviewer to detect spammers. They extracted the IP address of the reviewers to track the spammers' geo location. If a group of reviews in a specific time window is posted from a specific location, then the reviews are considered spam. The time stamps of the reviews are also a good indication of the spammers' activity [19]. Since some of the fake reviewers post opinions as soon as the product is released or some even before that, Fei et al. used the trait of the burstiness of the reviews in specific unexpected times to classify spam and ham [4].

2.2.2 Machine Learning

Machine learning methods have been mostly utilized in the past studies for fake review classification. These methods are categorized as supervised [19], unsupervised [19], and semi-supervised learning.

2.2.2.1 Supervised Learning

Cardoso et al. [3] presented a comprehensive analysis using supervised methods that utilize textual features. They employed different supervised classification algorithms including multinomial naïve Bayes (MNB), Bernoulli naïve Bayes (BNB), k-nearest neighbor (k-NN), decision tree (DT), RF, Rocchio, support vector machine (SVM) and MDLText. The performance of existing state-of-art methods vary due to the different engineered features, sentiment of the reviews, domain and the datasets used in the validation phase. The classification performance changed for all the mentioned classifiers when trained and tested differently, whether the reviews had positive or neg-

ative polarity, involved single or various domains, or were tested on different datasets.

2.2.2.2 Unsupervised Learning

When labeled datasets are not available, unsupervised classifiers are considered very helpful. Using real-world datasets provided by Amazon, Liu et al. proposed a unified unsupervised framework to detect fake reviews [10]. They used the idea that fake reviewers, usually, have not purchased or used the product they are describing. They are mostly reviewing for financial gain or for changing the reputation of a product. As such, their reviews would show abnormalities and deviation from expected values on many dimensions. The authors proposed the method of Review Deviation based Model RDM2. Their evaluation and analysis resulted in accuracy of 71.18% to 78.62% on different datasets from Amazon. Even though that method showed fair performance, it would need further investigation to consider some deviations that may wrongly affect the results. These deviations are caused by technical terminologies used by some honest experts which regular consumers might not use in their reviews.

2.2.2.3 Semi-Supervised Learning

In general, supervised learning needs a sizable set of labeled data for training. Providing this amount of labeled data in many applications including spam review detection is very time consuming and costly. However, providing a small set of labeled data is not as costly, and can significantly improve the performance of a classifier as it learns from the data. Such forms of semi-supervised learning methods have also been used for spam review detection.

A two-view method on review and reviewer features was used by Li et al. [8] to label a huge amount of unlabeled data using a small training set. Using this method there was finally enough labeled fake reviews to train a classifier. These reviews were labeled fake only if both views had regarded them as fake. Gómez and Rosso [5] used a different approach for fake review detection by implementing "PU" learning. It uses iterative trials of negative classification outcomes. Having removed the positive

classified instances, the model was trained on the rest of the unlabeled data and achieved a performance of 83.7% on the F-measure.

2.3 Proposed Method

In this section, we describe the dataset used, the pre-processing task performed on the raw dataset, and the proposed method that uses SOM and CNN for spam review detection. This combination of machine learning algorithms was first used by Fatima et al. [14] to predict cancer sub-types and stages, yielding very good results on cancer prediction. As in most data analysis experiments, the data needs to be pre-processed based on the needs for the specific methods to be used. As such, We cleaned and embedded words from the vocabulary used in all reviews of the corresponding dataset. Both Word2Vec and GloVe were used for embedding, and also word frequency and TF-IDF were utilized as the features for supervised training.

The proposed method first clusters semantically-related words present in the review corpus to a two-dimensional grid via a SOM training step. The map is then used to represent each review as a unique fixed-size image with different representation layers. Afterwards, the reconstructed review images are fed to a CNN to train and classify reviews as ham or spam. Our approach utilizes linguistic features to extract lexical diversities spread in ham or spam reviews.

2.3.1 Datasets

We tested our method on commonly-used benchmark datasets, one proposed by Ott et al. [16, 15] and the other proposed by Jiwei et al.[9]. The dataset proposed by Ott et al., which we call Single-domain, is a repository that contains 1,600 hotel reviews. This dataset contains 800 truthful (ham) reviews of both positive and negative polarity, 400 each, and 800 spam reviews, 400 for each polarity as well. The truthful reviews were collected from the TripAdvisor website, and correspond to the 20 most popular hotels in Chicago. Ott et al. recruited a group of people from Amazon Mechanical Turck (AMT) to write fake reviews for the same hotels. The lexical diversity

Reviews	Ham	Spam	All	Ham Only	Spam Only	Common
Unique words count	6,355	7,214	9,604	2,390	3,249	3,965

Table 2.3.1: Lexical diversity in the Single-domain dataset. Summary of the unique word counts extracted from a corpus of 1,600 cleaned reviews. A significant number of unique words used only in the ham or spam reviews can be utilized as features for the machine learning algorithms.

of the ham and spam reviews for this dataset are summarized in Table 2.3.1. It can be deducted from the table that the unique words used only in each ham or spam reviews can also be used as features learned by the machines. CNNs for instance can be used to extract these unique features for classification, yielding superior results.

The second gold standard dataset we used is comprised of reviews from three domains of doctors, hotels and restaurants; we call it Multi-domain. Jiwei et al. collected these data from three different groups of reviewers; the first group of reviews was written by real customers, the second group was written by the AMTs, and the third group was written by doctors, hotels and restaurant employees who are domain experts. Some statistics for this dataset are summarized in Table 2.3.2. We selected this dataset to frame and test the effectiveness of our proposed method on a multi-domain scenario, and to compare the performance of the method with that of a single-domain dataset.

Reviews	Ham	Spam	All	Ham Only	Spam Only	Common
Unique words count	8,986	9,124	12,728	3,604	3,742	5,382

Table 2.3.2: Lexical diversity in the Multi-domain dataset. Summary of the unique word counts extracted from a corpus of 2,840 cleaned reviews. Compared to the Single-domain dataset, the Multi-domain dataset includes doctors and restaurants reviews in addition to the hotels reviews, as well as some fake reviews written by domain experts in addition to AMT fake reviews, which are not done by domain experts.

2.3.2 Pre-processing

In order to classify reviews as fake or truthful, we first performed some pre-processing steps to make the data ready for the classifier we used. The pre-processing steps we used are described below.

Reviews were first cleaned by removing all the punctuation and special characters, such as "(", ",", "*", and other special characters. Then, all the words were converted to lower case and tokenized as unigram terms. Tokenized words were used to calculate the Term Frequency-Inverse Document Frequency (TF-IDF) of the words and fetch their embedding vectors from the pre-trained dictionary, as described below.

The TF-IDF weight is a statistical measure used to evaluate how important a word is to a document in a collection of texts or corpus. The importance increases proportionally to the number of times a word appears in the document. On the other hand, the importance decreases by the frequency of the word in the corpus. Rare words contribute more weights to the model, whereas highly frequent words would not provide much information gain.

Word embedding is a technique used to represent Natural Language in a fixed-dimension vector of real numbers. This technique automatically captures the essence and relationships among words by employing deep learning. The main idea is that the meaning of a word is given by the words that frequently appear close-by in the text. In our experiments, we have tested two well-known pre-trained word embedding dictionaries, namely Word2Vec and GloVe, and selected the one that performed the best.

Word to Vec (Word2Vec) embedding dictionary includes three million words and phrases out of roughly 100 billion words from the Google News dataset [11]. Each word, represented as a 300-dimensional vector, is created by a deep learning model that computes and measures how well a certain word can predict its surrounding words. In this dictionary, some of the stop words such as "a", "and", "of" are excluded, while others such as "the", "also", and "should" are included. The dictionary includes misspellings of words and commonly paired words, i.e., "Soviet_Union" and

"New_York".

Global Vectors (GloVe) is a word representation dictionary developed at Stanford University with 400,000-word vocabulary trained on a six billion token corpus from Wikipedia 2014 + Gigaword 5 [17]. GloVe provides pre-trained word vectors of 50, 100, 200 and 300 dimensions. Unlike Word2Vec, which only considers local contexts, GloVe captures the meanings in the vector space and takes global count statistics into account.

2.3.3 Detection of Spam Reviews

We have designed an approach that is used to distinguish between spam and ham reviews. The method is based on SOM and CNN, which has been successfully applied to predict cancer subtypes on multi-omic data [14]. In our approach, we take advantage of the strengths for each method: the SOM allows for representation learning and dimensionality reduction, while the CNN captures the spatial, hidden relations embedded in the features, which are useful for image classification. Since it is almost impossible to figure out the relationships in the input vectors in such a high-dimensional space, the SOM helps represent the data onto a lower dimension, typically, a two-dimensional map, while preserving the relationships of the input data in the lower-dimensional space. The data can then be distinguishable by the CNN provided that an image representation is used. The main steps carried out by the proposed method are described below.

2.3.3.1 Step 1: Constructing the SOM

The procedure for this step is depicted in Figure 2.3.1. Each review is pre-processed and the words used are tabulated in different lists. Then, the unique words from all reviews' lists are tabulated in a vocabulary list, $V_{i\ i=1\rightarrow m}$. The embedding matrix, $E_{ij\ i=1\rightarrow m,\ j=1\rightarrow q}$, is subsequently constructed. Each row of the matrix represents the corresponding word embedding vector. The vector is fetched from a pre-trained embedding dictionary of fixed size q such as Word2Vec or GloVe. A SOM, namely an

arbitrarily sized grid map of cells, $C_{ij\ i=1\to n,\ j=1\to n}$, is then used to cluster all words in the vocabulary using an unsupervised learning algorithm. Clustering happens as each word, represented by its equivalent embedding vector, is assigned to a cell of the SOM grid during the training process.

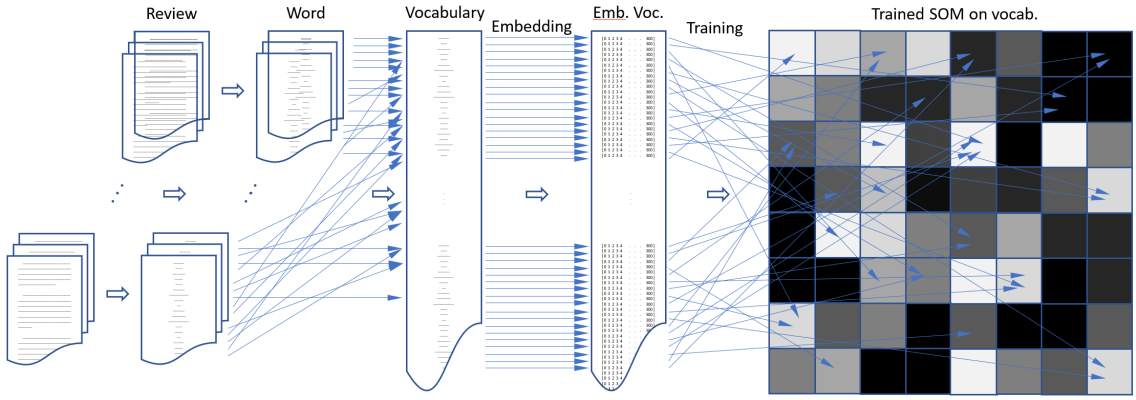

Fig. 2.3.1: Step 1: Construction of the SOM is performed by putting all the unique words in all reviews in a vocabulary list. Word2Vec or GloVe embedding is consequently employed to represent each word as a fixed length vector. Self organizing map is used then to cluster all the words into cells of the SOM grid.

Each cell of the SOM grid is a randomly initialized weight vector, $W_{ij\ i=1\to n,\ j=1\to n}$, of the same size as the word embedding, $W_{ij} = [w_{ij,1} \quad w_{ij,2} \quad ... \quad w_{ij,q}] \quad i = 1 \to n,\ j = 1 \to n.$

In the training phase, each word in the vocabulary, V_k, is examined to determine which cell, C_{ij}, is the closest to the word embedding vector, based on Euclidean distance d_{ij}:

$$V_k \in C_{ij} \quad | \quad D = Min(d_{ij}) \quad , \tag{1}$$

$$d_{ij} = \| V_k - W_{ij} \| = \sqrt{\sum_{l=1}^{q} (V_{kl} - W_{ij,l})^2} \quad , \tag{2}$$

where $i = 1 \to n,\ j = 1 \to n.$

The word is assigned to that SOM cell, also referred to as the *best matching unit* (BMU). The weights of the BMU and its close-by cells are then updated and, as a result, they become situated near the corresponding word (embedding vector) as

follows:

$$W(t+1) = W(t) + \Theta(t)L(t)[V(t) - W(t)]\,, \tag{3}$$

where $t = 1, 2, \ldots, s$ is the iteration of maximum at s. Θ and L are the neighborhood function and the learning rate respectively, which are both decaying values over increasing iterations. These influence how intensely the weights of the BMU and neighboring cells become gravitated towards the word embedding vector:

$$L(t) = L_o e^{-t/\lambda}\,, \tag{4}$$

$$\Theta(t) = e^{-d^2/2\sigma^2(t)}\,, \tag{5}$$

$$\sigma(t) = \sigma_o e^{-t/\lambda}\,, \tag{6}$$

where d is the distance between a BMU cell and its neighboring cells, and λ is a configurable decay constant. The initial values of L_o and σ_o are configurable parameters for the training phase. The last equation above is particularly interesting as it shows not only that neighboring cell weights are updated much less along the training, but also that the number of cells considered as neighbors is reduced as the neighborhood radius decreases.

At the end of the training, each word of the vocabulary is assigned to a SOM cell. That does not necessarily mean that all SOM cells contain words assigned to; there maybe some cells without any assigned word. That likelihood increases with the size of the SOM grid selected, n.

The number of words assigned to a single SOM cell is a measure of how populated a cluster of words is. At the same time, that alone may not show how apart the words are from each other. Another measure often used is referred to as Mean Inter-neuron Distance (MID), which shows how scattered or close the cells themselves are based on the weights assigned to them. The normalized gray scale or color graph of this measure is usually provided for the trained SOMs. This measure, denoted by D in the

equation below, can be defined as the average or total sum of distances between a cell weight and the weights of its (usually eight) neighboring cells. The lower this value is, shown by darker gray cell in the graphs presented, the closer and more densely clustered the neighboring cells are, and so are the words assigned to these cells.

$$D_{ij} = \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} \| W_{ij} - W_{kl} \| \tag{7}$$

2.3.3.2 Step 2: Converting Reviews to Images Using the SOMs

The procedure for this step is shown in Figure 2.3.2. In this step, each review is converted into a single or multiple layered image or matrix. The image is of the same size as the SOM grid map constructed in Step 1. Each layer of the image corresponds to a different feature of the reviews.

The density of each pixel for a layer is the sum of the quantized measure of the feature chosen for all the words associated with that pixel. We used the number of distinct words in a review as the feature selected for the first layer of the image. TF-IDF of the unique words used in the review was used as the feature for the second layer of the image.

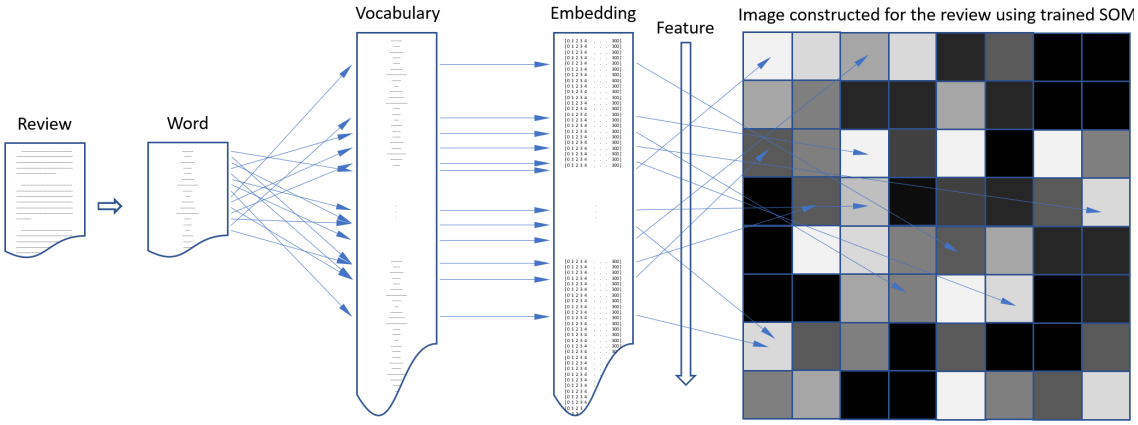

Fig. 2.3.2: Step 2: Conversion of each review to an images is performed by finding each word of the review in the vocabulary and its corresponding embedding vector. The assigned cell (pixel) of the SOM grid (image) is retrieved from the previous step. The intensity of the pixel, initially zero, is added to the value of the word feature considered for that layer of the image.

2.3.3.3 Step 3: Training the CNN

The procedure followed in this step is depicted in Figure 2.3.3. The images constructed in Step 2, which represent the reviews, are fed to a CNN for the supervised training and then classification. The CNN is a well-known supervised deep learning algorithm that detects and learns the hidden attributes of the training data, especially images. Thus, the CNN is an excellent choice to classify the generated review images using their corresponding labels provided in the dataset.

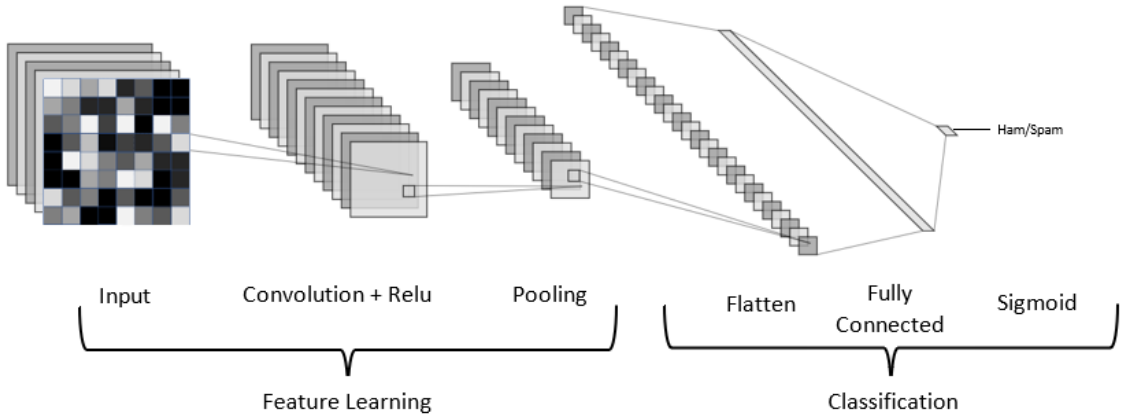

Fig. 2.3.3: Step 3: CNN is employed to learn the relation between the images produced in Step 2, and the labels for each review in the dataset. The CNN used in this work contains layers for convolution, pooling, and flattening, and are fully-connected to the output layer.

The CNN architecture used in this work is a stack of convolution, pooling, flattening, fully connected and a dropout layer. In this architecture, we apply 32 filters of size 3x3 in the convolution layer, followed by an activation function of ReLU. Next, we apply a sub-sampling (Max-pooling) layer of 2×2 to make the feature maps smaller and more manageable. The flattening layer is applied to reshape the structure of the representations obtained from the pooling layer. In the first fully connected layer, we assign 100 hidden neuron to reach the classification decision. Consequently, we apply dropout to prevent the network from co-adaptation of the features. At the end, we apply a fully connected output layer that uses the Sigmoid function to output the final probabilities for each class.

2.3.3.4 Step 4: Classification of New Reviews

Once both the SOM and the CNN used in our method have been constructed and trained, a new review can be classified using the procedure depicted in Figure 2.3.4. This classification is also used for evaluating the classifier through training and testing accuracy.

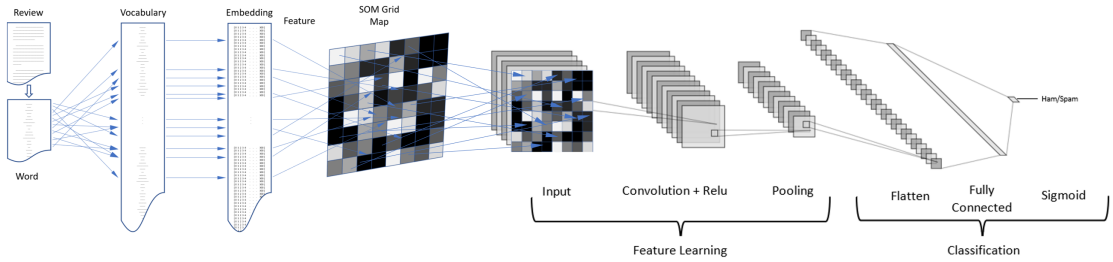

Fig. 2.3.4: Step 4: A new review can be classified using the trained SOM and CNN described in Steps 2 and 3 respectively. The output image of the trained SOM is used as the input of the trained CNN classifier.

First, the new review is converted into a (multilayer) image in the same way as described in Step 2 using the trained SOM. The resulting image is classified using the trained CNN obtained in Step 3.

2.4 Results and Discussion

A number of experiments have been conducted to evaluate the performance of the proposed method. We performed our experiments using Python language in Google Colab-Pro environment and set the runtime to use GPUs. For SOM deployment, we used the Minisom version 1.1.2 library [24] to build the cell grid map and to cluster words utilizing the vocabulary embedding matrix. For CNN deployment, we used Tensorflow, a high-level API of "tensorflow.keras" to build the CNN and train it. Tensorflow version 2.2.0 was used in our work, which was the Google Colab default at the time we conducted our experiments.

Through the set of experiments for performance evaluation and comparison, we tested our method with two different embedding techniques. We used pre-trained

word embedding dictionaries of both Word2Vec and GloVe, in four different dimensions of 50, 100, 200 and 300, to fetch the corresponding word vectors for our vocabulary. To find the best SOM hyperparameters to use, we employed a grid search on different cells map sizes and the neighbourhood radii. SOM map sizes of 20, 30, ... , 100 squared and neighborhood radii of 1, 2, 3, 4, 5, 7 and 9 were tested to observe the effects in performance.

Here, we provide the results of the SOM training in Figure 2.4.1 for illustration. This is for the case of SOM grid size of 20×20, neighboring function starting at 5 and GloVe-300 embedding used. The figure shows unsupervised learning and clustering of all words in the vocabulary, and hence it does not have anything to do with the truthfulness of a review, or whether or not the review is spam. The purpose of this illustration is to show that the CNN can learn patterns of association of a word with probability to occur in spam or ham reviews, and use it to classify a new review as spam or ham. Green cells contain words appearing in ham reviews only, while red cells contain words appearing in spam reviews only; the black cells do not contain any words; the other cells which are colored with the mixture of red and green contain words from both ham and spam reviews. Some of the clusters are enclosed with blue and purple for the words used in ham and spam reviews respectively. We have also selected two review images constructed based on the SOM, one from spam reviews and another from the truthful ones, shown in Figure 2.4.2. The different sizes of green and red circles inside the grid cells represent the number of words from one review assigned to each cell.

We deployed two CNN architectures and selected the one resulting the best performance as reported in Section 3.3.3. The Keras Tuner framework was also used to test and find the best hyperparameters for our model. The CNN architecture chosen still provided better results. Furthermore, we implemented the early stopping mechanism to avoid over-fitting and under-fitting along the training data. This mechanism stops the training at the point in which the performance of the validation set starts to degrade. We used the following metrics: Accuracy, Precision, Recall and F1-Score to evaluate the performance of the method for different combinations [4]. Also, the

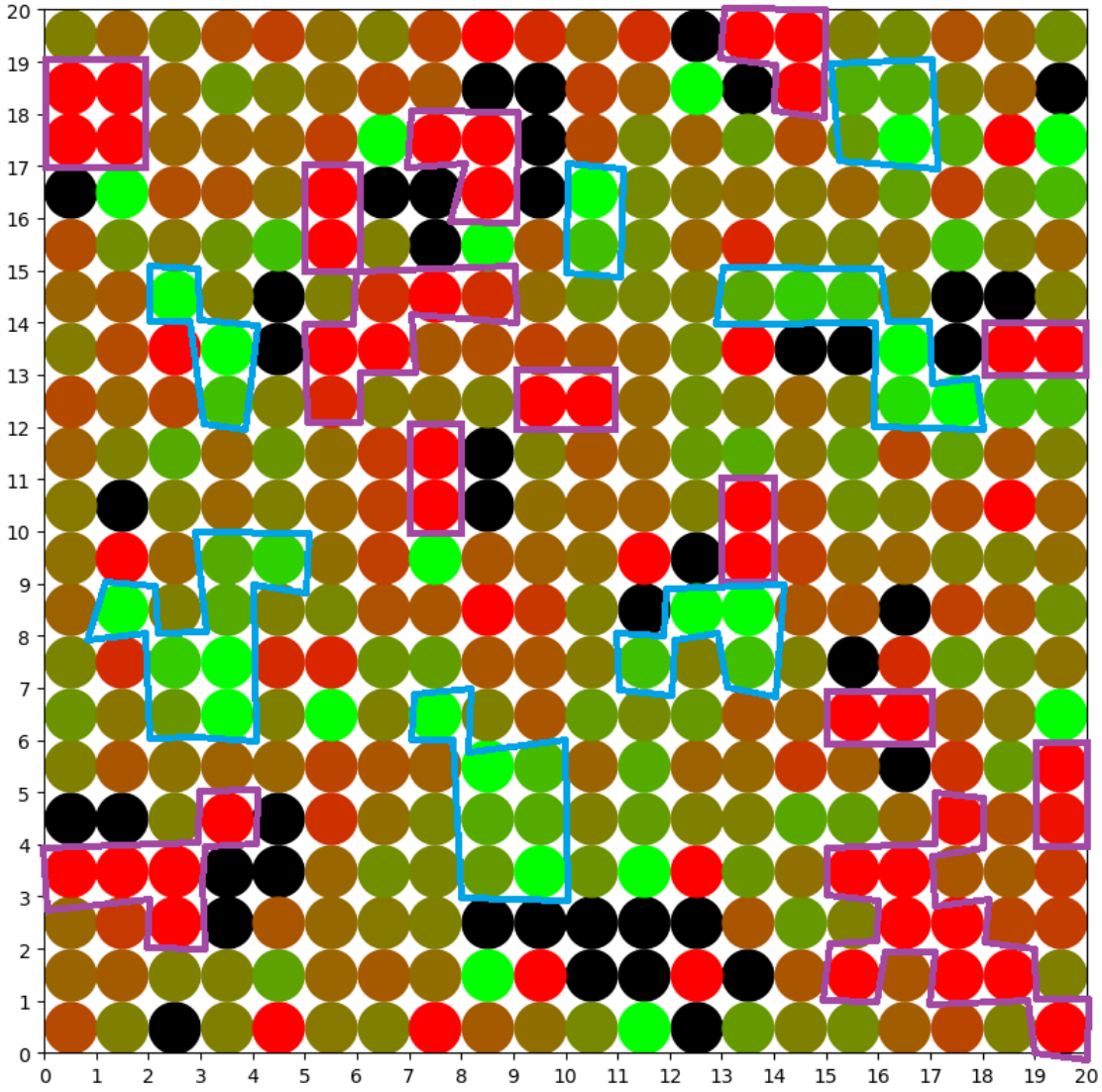

Fig. 2.4.1: Clustering of the vocabulary in a trained SOM.

CNN was trained by splitting the data into 80% for training and 20% for testing. To make sure the results obtained were not biased toward a special case, we repeated the same tests using 10-fold cross validation.

We summarize the results after applying the proposed approach on the Single-domain dataset, described in Section 2.4.1, and the results on the Multi-domain dataset, described in section 2.4.2. The comparative results between our proposed

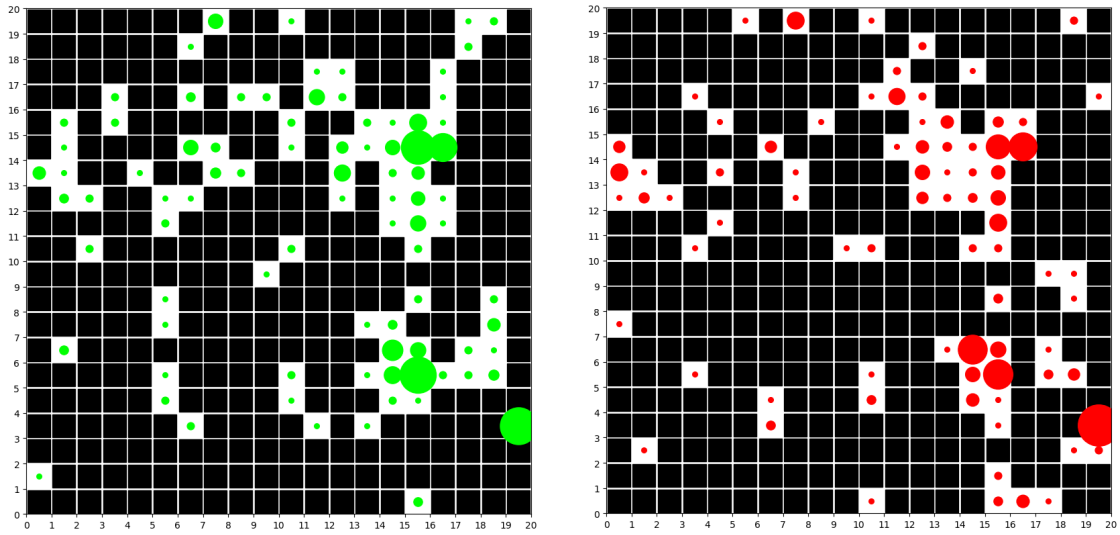

Fig. 2.4.2: Distribution of the words in ham (left) and a spam (right) reviews into a 20×20 SOM grid. The size of the circles show the proportion of words associated with each cell.

method and the other approaches are discussed in Section 2.4.3.

2.4.1 Single-domain Dataset

Figure 2.4.3 shows the performance of our method using GloVe-300 embedding. The effects of the size of the SOM grid and neighborhood radii on the method performance, validation accuracy in this case, are shown in the figure. We also performed similar tests using 10-fold cross validation. This was done to make sure the first set of results were not representing a special case. The test results for the 10-fold cross validation are summarized in Figure 2.4.4. In general terms, we deduct that the higher the size of the SOM grid, the better our method performs. Similarly, smaller neighborhood values result in better performance.

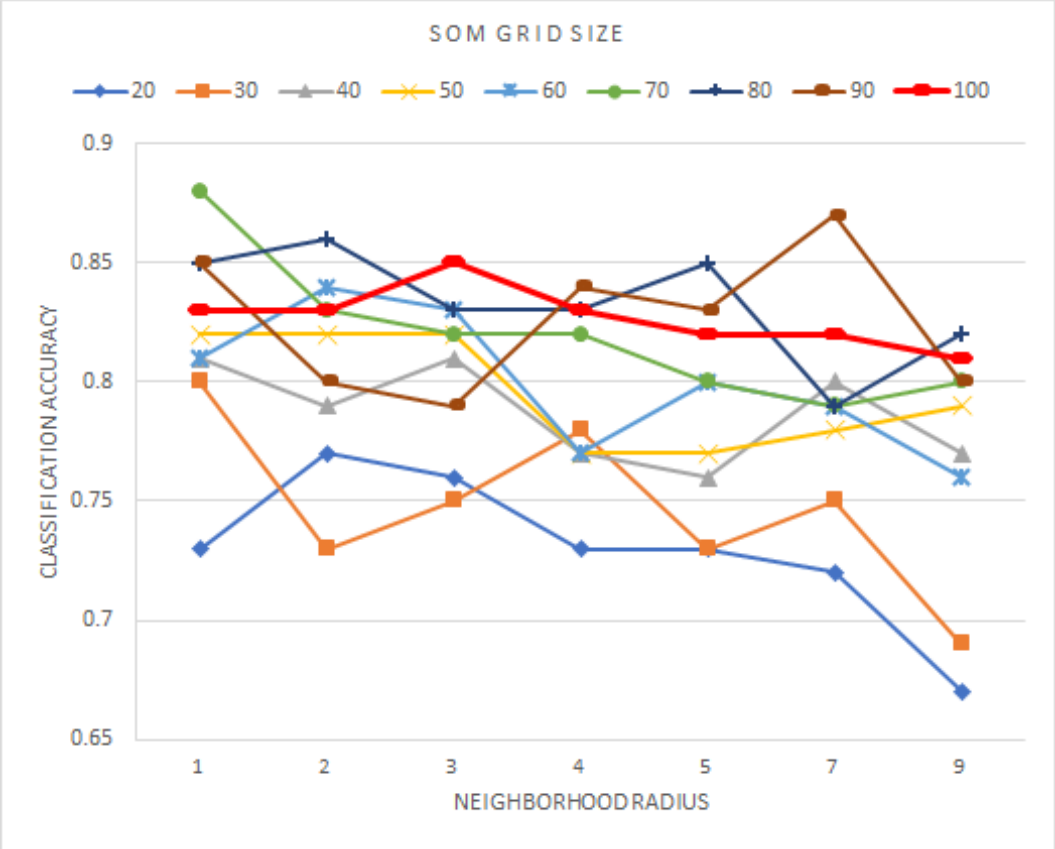

Fig. 2.4.3: The effect of SOM grid size and neighborhood radius on the performance using GloVe-300.

Observing the results, higher accuracy for a larger SOM grid can be attributed to the effect on the resolution of the map. When the size of the grid map is smaller, the SOM forcefully places less semantically closer words into the same cluster of cells. In contrast, when the grid map is larger, the SOM finds more room to cluster words that are semantically closer, or equivalently similar on a higher-dimensional space. Since we used these grid maps to create the review images and feed them to the CNN for the classification, the more precise and detailed these images are constructed, the better CNN learns their hidden attributes, which subsequently produces better classification as ham or spam.

Similarly, the neighborhood sigma values are related to the dimension of the SOM grid maps. When the dimension of the grid map is small, similar words are densely distributed among the cells. Thus, the smaller the value of sigma is, the more accu-

rate of the model is, since the SOM clusters words in a more compact manner. In contrast, when the SOM grid map is larger, similar words are more widely distributed among neighboring cells. Therefore, not only smaller values of sigma result in higher accuracy, but also higher values of sigma lead to good results.

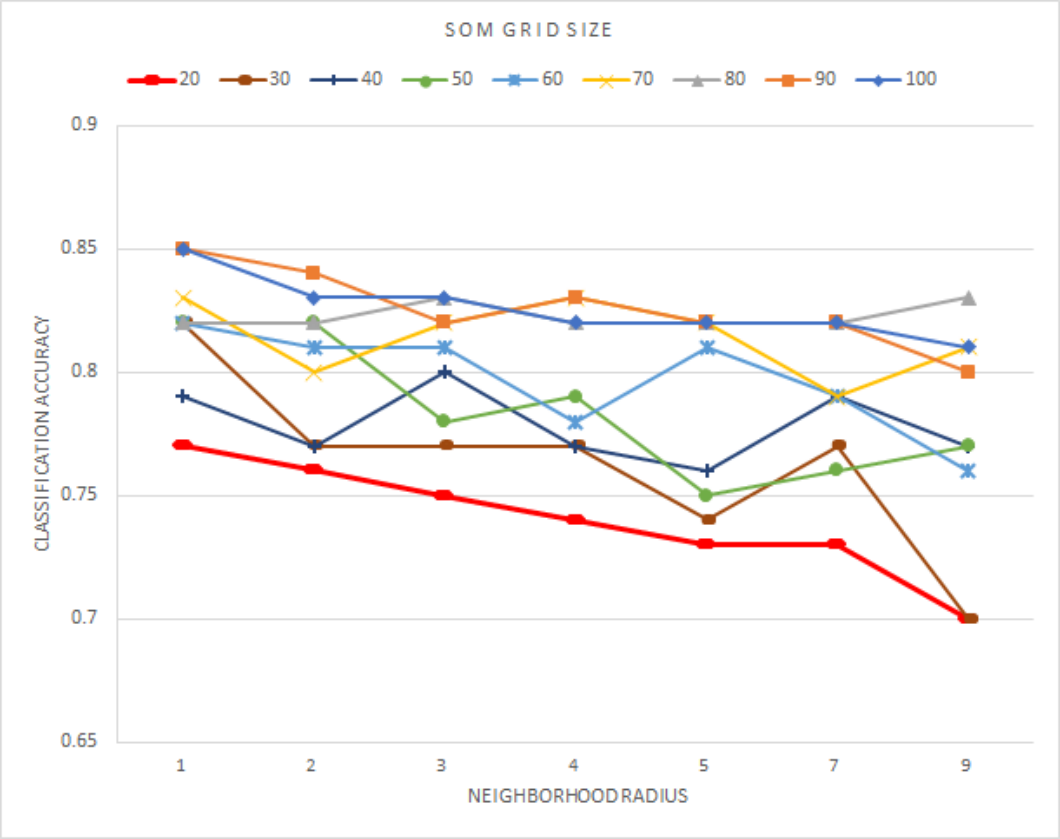

Fig. 2.4.4: The effect of SOM grid size and neighborhood radius on the performance using GloVe-300 validated by 10-fold cross validation

The results for the same experiments using Word2Vec embedding are summarized in Figures 2.4.5 and 2.4.6. While the performance measure is not as high as when using GloVe-300, we observe similar behavior here.

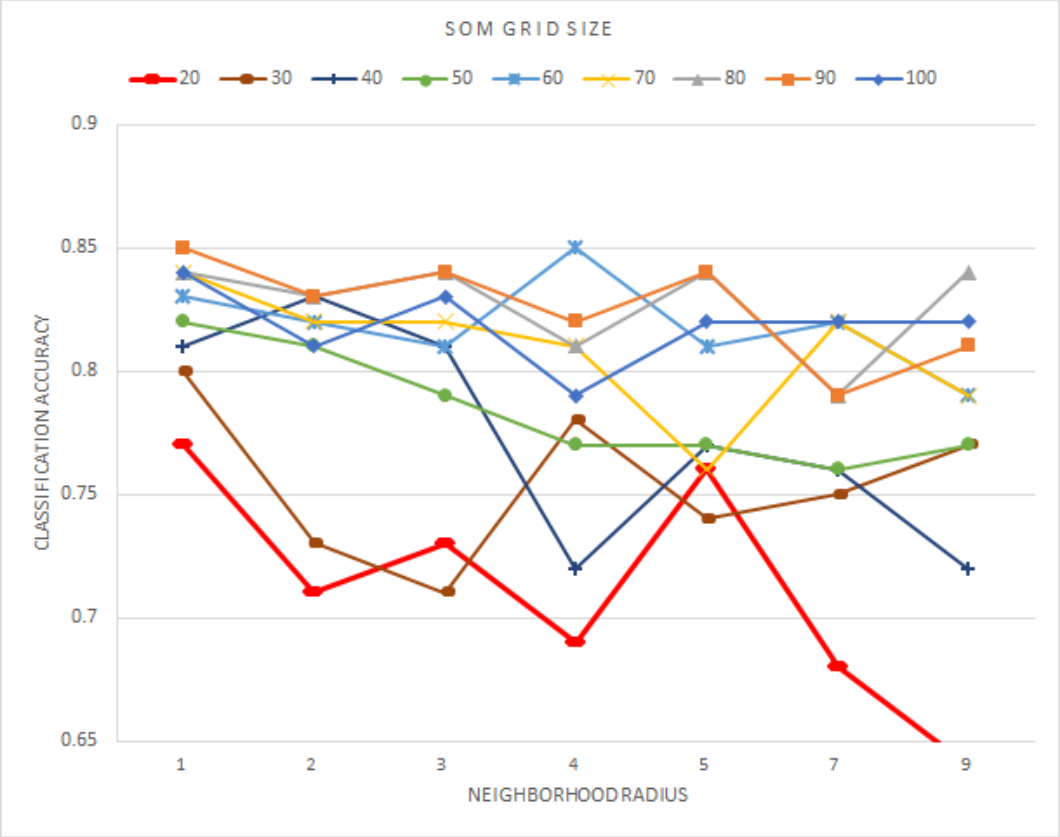

Fig. 2.4.5: The effect of SOM grid size and neighborhood radius on the performance using Word2Vec

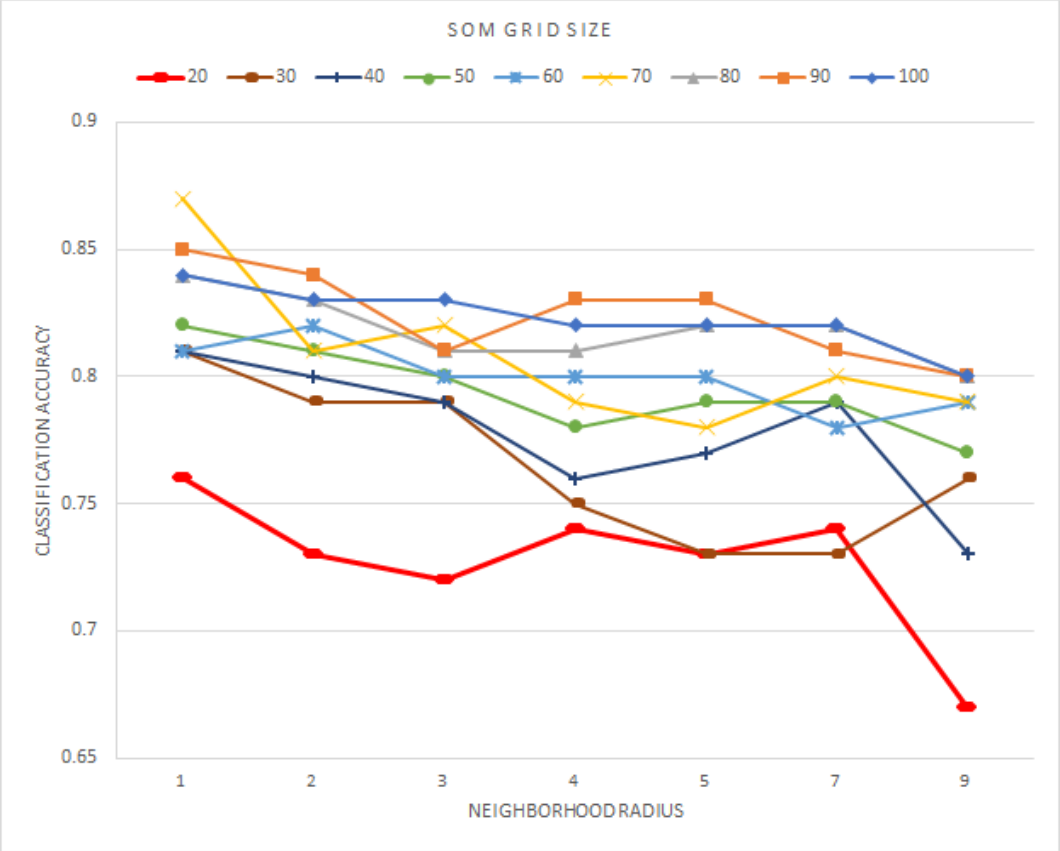

Fig. 2.4.6: The effect of SOM grid size and neighborhood radius on the performance using Word2Vec validated by 10-fold cross validation

While we have tested different embedding methods of Word2Vec, GloVe-50, GloVe-100, GloVe-200 and GloVe-300, we provide the results for the same 300-dimension Word2Vec and GloVe. In Figure 2.4.7, we show the effect of the embedding method on the performance of the classification, evaluated via different metrics. GloVe-300 yields the best results in most of the cases. In contrast, choosing the second best can marginally depend on the performance metric chosen.

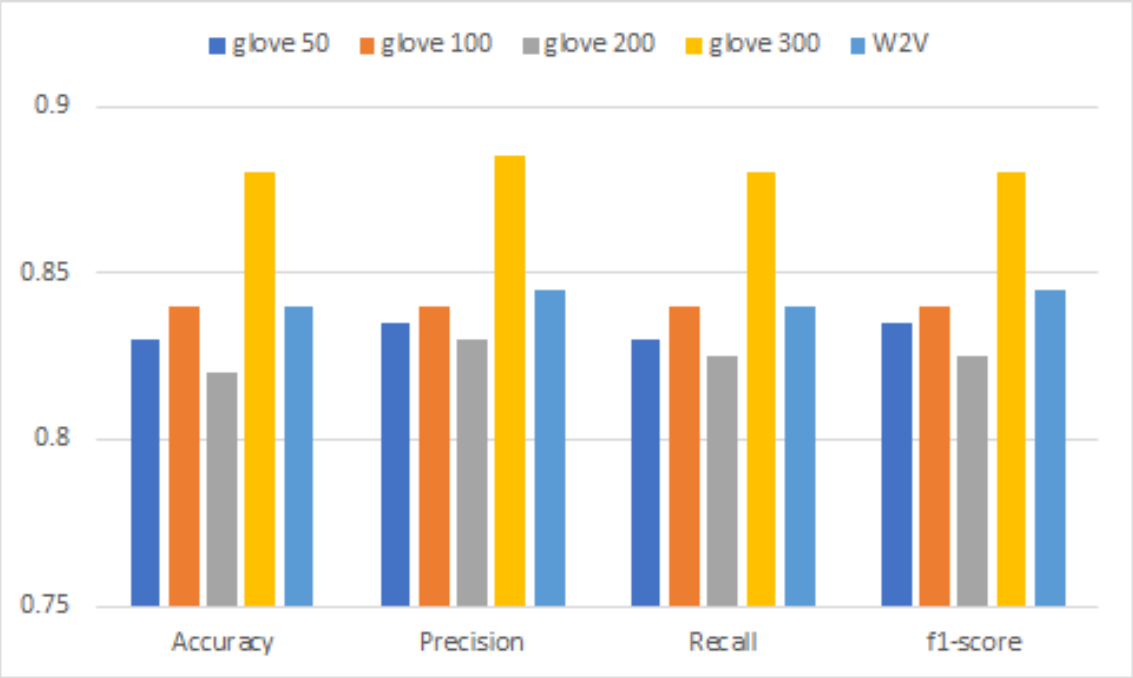

Fig. 2.4.7: The effect of embedding methods on the performance metrics for SOM size of 70×70 and neighborhood function of 1, the highest accuracy reached in our experiments

2.4.2 Multi-domain Dataset

Similar to the Single-domain dataset, Figure 2.4.8 shows the performance of our method using GloVe-300 embedding on the Multi-domain dataset. The effect of the SOM grid size and neighborhood radii on the method test accuracy are shown in the figure. For the Multi-domain dataset, we observe the same trend as that of the Single-domain dataset.

We have also explored the performance of the proposed method on the Multi-domain dataset as a whole and separated as individual domains. The metrics are shown in Table 2.4.1, corroborating the effectiveness of the proposed method on a multi-domain dataset. The best performance we observe is the one obtained through grid search, which uses a map size of 70 and a sigma radius of 2. The review images were randomly split into training and testing sets with a ratio of 80:20. Using these best SOM parameters, we further applied grid search to tune the other hyperprameters. For the Multi-domain dataset, we utilized the Keras functional API in order

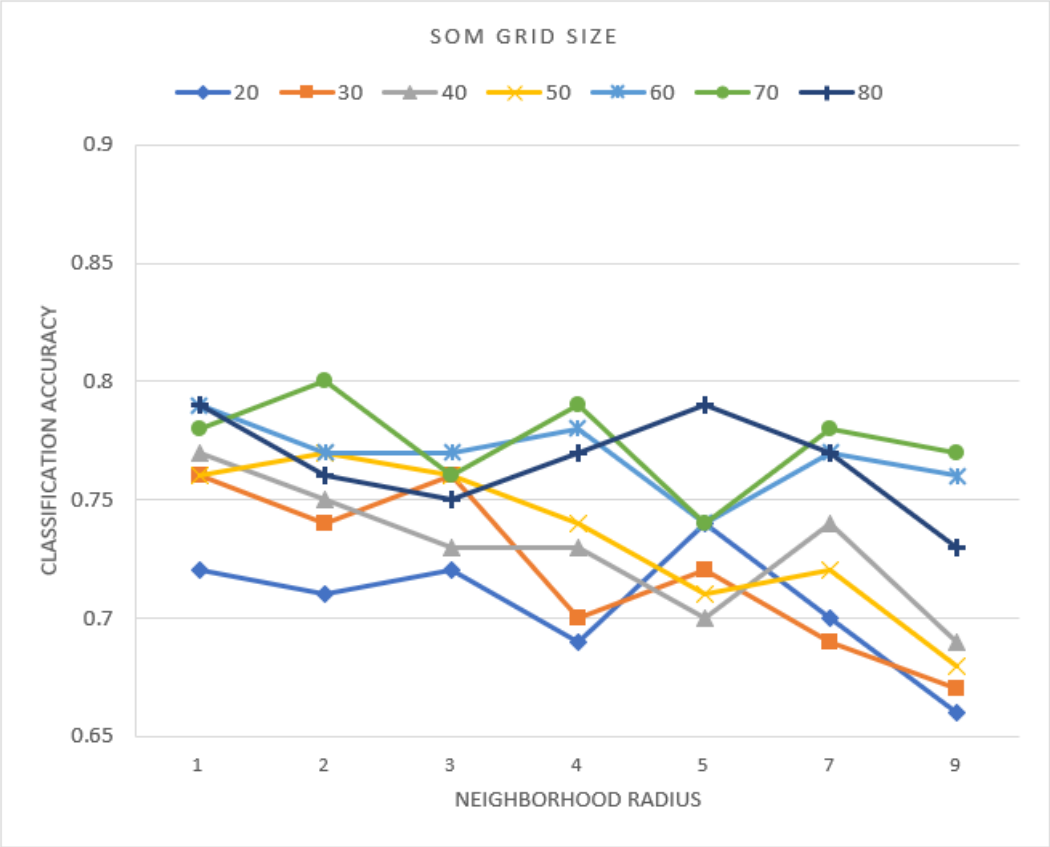

Fig. 2.4.8: Effect of the SOM grid size and neighborhood radii on the performance on the Multi-domain dataset.

to obtain multiple inputs; the first inputs to a CNN layer and the second inputs to a dense Neural Network (NN). Further, the output of the both networks were combined using a dense layer. This resulted in a boosted accuracy of 0.82; the results listed the last row of Table 2.4.1. Five-fold cross validation was also applied and the results are shown in Table 2.4.2.

2.4.3 Comparison with Other Methods

Ren and Ji [25] used the same Multi-domain dataset to test the performance of their method. They reported the performance of various neural network-based methods and compared them with their proposed method, Bi-directional Average GRNN. They used AMTs and customer reviews and split the data into 80 percent for training and 20 percent for testing and validation. Here, we provide their results and compared them

	Accuracy	Precision	Recall	F1-score
Hotel	0.86	0.85	0.86	0.85
Doctor	0.94	0.93	0.93	0.93
Restaurant	0.88	0.89	0.87	0.88
Multi-domain	0.82	0.82	0.81	0.82

Table 2.4.1: Performance metrics for SOM-CNN on the Multi-domain dataset.

	Accuracy	Precision	Recall	F1-score
Hotel	0.82	0.82	0.82	0.82
Doctor	0.85	0.84	0.83	0.83
Restaurant	0.83	0.83	0.83	0.83
Multi-domain	0.80	0.80	0.79	0.80

Table 2.4.2: Performance metrics for SOM-CNN on the Multi-domain dataset by applying five-fold cross-validation.

with those of our method, SOM-CNN, in Table 2.4.3. To make a fair comparison, we used the same types and number of reviews, as mentioned above. The results show that our SOM-CNN model yields better accuracy than the best performing methods for spam review classification, on Multi-domain contexts.

Method	Accuracy	Macro-F1
Average	0.730	0.739
CNN	0.759	0.774
RNN	0.632	0.648
GRNN	0.790	0.799
Average GRNN	0.801	0.807
Bi-directional average GRNN	0.836	0.834
Le and Mikolov [2014]	0.761	0.776
SOM-CNN	0.871	0.870

Table 2.4.3: Performance comparison of different methods for the Multi-domain dataset.

References

[1] Nga N Ho-Dac, Stephen J Carson, and William L Moore. "The effects of positive and negative online customer reviews: do brand strength and category maturity matter?" In: *Journal of Marketing* 77.6 (2013), pp. 37–53.

[2] Neelam Duhan, Mamta Mittal, et al. "Opinion mining using ontological spam detection". In: (2017), pp. 557–562.

[3] Tiago A. Almeida Emerson F. Cardoso Renato M. Silva. "Towards automatic filtering of fake reviews". In: *Neurocomputing* 309 (2018), pp. 106–116.

[4] Geli Fei et al. "Exploiting Burstiness in Reviews for Review Spammer Detection". In: *ICWSM* (2013).

[5] Donato Hernández Fusilier et al. "Using PU-learning to detect deceptive opinion spam". In: (2013), pp. 38–45.

[6] Naveed Hussain et al. "Spam Review Detection Using the Linguistic and Spammer Behavioral Methods". In: *IEEE Access* 8 (2020), pp. 53801–53816.

[7] Nitin Jindal and Bing Liu. "Analyzing and detecting review spam". In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE. 2007, pp. 547–552.

[8] Fangtao Huang Li et al. "Learning to identify review spam". In: (2011).

[9] Jiwei Li et al. "Towards a general rule for identifying deceptive opinion spam". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2014, pp. 1566–1576.

[10] Yuanchao Liu and Bo Pang. "A unified framework for detecting author spamicity by modeling review deviation". In: *Expert Systems with Applications* 112 (2018), pp. 148–155. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.06.028. URL: http://www.sciencedirect.com/science/article/pii/S0957417418303749.

[11] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[12] Arjun Mukherjee et al. "Spotting opinion spammers using behavioral footprints". In: (2013), pp. 632–640.

[13] Asghar Nabiha. "Yelp Dataset Challenge: Review Rating Prediction". eng. In: Ithaca: Cornell University Library, arXiv.org, 2016. URL: http://search.proquest.com/docview/2079845554/.

[14] Luis Rueda Nazia Fatima. "iSOM-GSN: An Integrative Approach for Transforming Multi-omic Data into Gene Similarity Networks via Self-organizing Maps". In: *Bioinformatics* nmbr (2020), pp. 1367–4803.

[15] Myle Ott, Claire Cardie, and Jeffrey T Hancock. "Negative deceptive opinion spam". In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: human language technologies*. 2013, pp. 497–501.

[16] Myle Ott et al. "Finding deceptive opinion spam by any stretch of the imagination". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics. 2011, pp. 309–319.

[17] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[18] Jyoti Prakash Singh et al. "Predicting the helpfulness of online consumer reviews". In: *Bus Res* 70 (2017), pp. 346–355.

[19] Shebuti Rayana and Leman Akoglu. "Collective opinion spam detection: Bridging review networks and metadata". In: (2015), pp. 985–994.

[20] Shivani Saini, Sunil Saumya, and Jyoti Prakash Singh. "Sequential purchase recommendation system for e-commerce sites". In: *IFIP international conference on computer information systems and industria management* (2017), pp. 366–375.

[21] Sunil Saumya and Jyoti Prakash Singh. "Detection of spam reviews: a sentiment analysis approach". In: *Csi Transactions on ICT* 6.2 (2018), pp. 137–148.

[22] Sunil Saumya et al. "Ranking online consumer reviews". In: *Electronic Commerce Research and Applications* 29 (2018), pp. 78–89. ISSN: 1567-4223. DOI: https://doi.org/10.1016/j.elerap.2018.03.008.

[23] GM Shahariar et al. "Spam Review Detection Using Deep Learning". In: (2019), pp. 0027–0033.

[24] Giuseppe Vettigli. *MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map*. https://github.com/JustGlowing/minisom/.

[25] Ren Yafeng and Ji Donghong. "Neural networks for deceptive opinion spam detection: An empirical study". In: *Information Sciences* 385 (2017), pp. 213–224.

[26] Cennet Merve Yilmaz and Ahmet Onur Durahim. "SPR2EP: A Semi-Supervised Spam Review Detection Framework". In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2018, pp. 306–313.

[27] Feng Zhu and Xiaoquan Zhang. "Impact of online consumer reviews on sales: The moderating role of product and consumer characteristics". In: *Journal of marketing* 74.2 (2010), pp. 133–148.

CHAPTER 3

Conclusion and Future Work

3.1 Conclusion

We have introduced a new method used to distinguish spam reviews from genuine ones. The proposed framework has been applied on a well-known dataset taking into account contextual features from the body of the reviews. We used two different embedding techniques to investigate the effectiveness of our method and compare their performance. We combined unsupervised learning via a SOM to cluster semantically-similar words and create images for the reviews. The review images are then fed to a CNN for training and classification.

The results show that the model based on GloVe-300 embedding achieves the best performance. Our SOM-CNN method yields superior results on both single and multi-domain contexts. A careful observation and visual inspection of the results reveals that the performance of our method has a direct relation to the size of the SOM grid map and the neighborhood radii. We used two features, word density and TF-IDF, in the map cells as two layers of the review images being constructed.

3.2 Future Work

This work can be further extended as follows:

- Improving the performance of the proposed method through combining the proposed method with different NN algorithms such as RNN. RNN is known to capture the sentiment of the context. This might add to ability for detecting

spam reviews.

- Extracting new contextual based features and adding more layers of features to the review images created using SOM. For instance, Part of Speech (POS) Tagging can be used as a new feature, represented by an additional layer of the review images. More layers may provide additional information in the model for better classification.

- Compiling other features such as reviewer and metadata-based characteristics in conjunction with our proposed methods to boost the performance. These features cannot be represented by additional SOM generated layers. Rather, these features need to be processed through other methods. The result of these methods and the output of the proposed method can then be merged using functional APIs in the fully connected layer for the final classification.

- Applying our proposed method to other language applications, such as fake news detection. As spam review and fake news detection are very similar in nature any method that works in one of those might work on the other one as well.

VITA AUCTORIS

NAME:	Ashraf Neisari
PLACE OF BIRTH:	Khorramdareh, Zanjan, Iran
EDUCATION:	B.Sc. degree in Computer Science, Azad University, Abhar, Zanjan, Iran, 2012
	Computer Networking Technology Diploma, St. Clair College, Windsor, Ontario, Canada, 2017
	M.Sc. Computer Science, University of Windsor, Windsor, Ontario, Canada, 2020